適応的手法による船舶の運動制御

学位論文

題目 適応的手法による船舶の運動計算

指導教授 森下 敬宮

商船学研究科 [illegible] 専攻

平成 [illegible] 年入学

一貫制 3 年入学

氏名 下高原 [illegible]

平成
一貫制 3年 1月31日提出

適応的手法による船舶の運動制御

下高原　剛

目　　次

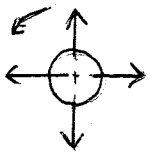
始めに

現在，船舶運行に対して省力化，高知能化の見地から，船舶自体の高度化が世界的に求められ，特に四方を海に囲まれた日本への期待は大きい．

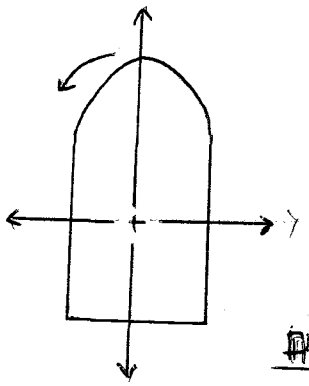
船舶の運行に際し，運航員が神経を使っている事は多くあると思われる．その一つは，出入港時であろうが，現在の多くの船舶は色々高度な機器が備えられ，例えばサイドスラスターがあり，これを利用することによってサイドスラスターの登場する以前より出入港時の操船が楽になった感もある．しかし出入港時の苦労は，大変なものである事に代わりは無い．

その理由の一つにサイドスラスターと言う便利な機械はあるが，それを制御（この場合，肉眼による計測，調節，動かしている）のは人間であり，ただでさえ，省力化で乗組員一人当りの負担が増えている所へ神経を使う出入港の際，サイドスラスター及びプロペラを人間がすべて制御し，何ら補助的な制御がなされて無いところにあると思われる．そこで，できるならサイドスラスター（プロペラ）の推力を計算機で統合的に制御して操船者の負担を軽減することが考えられる．

その方法として色々なアプローチがあると思うが，サイドスラスターの推力をブリッジの中から計算機を介してコントロールする事を考察してみたい．



コントローラ



船体運動

目的　1．サイドスラスター及びプロペラの推力を計算機による補助的な制御をすることによって，速度（例えば到達させたい速度だけ入力する）船体の移動（例えば船体を動かしたい距離だけ入力する）を容易にし，かつ操船者の負担を軽減する．

目的　2．インターフェイスとして船体の速度，移動量などの入力に際し統合的なマン＿マシンインターフェイスの優れた操船機構（例えばステイック状のコントロラー等）でコントロールを容易にすることによって，速度，船体の移動をスムーズに行い，かつ乗組員の負担を軽減する．

目的　3．最終的には，サイドスラスター及びプロペラの推力を高度に知能化されたメインコンピュータによって統合的に制御することにより入出港時等の船体の位置の微調節から，回頭時に至るまで幅広い船体の制御が可能となるようにする．

文献（１）を用いて簡単な伝達関数を求めることは，可能であるかもしれないが，船舶は自然界の中を移動する物であるので，常に文献（１）で求めた伝達関数である保証は無く，又低速時から高速時まで常に一定の伝達関数であるという保証も無い．

そのため今回は，伝達関数の不確かさや未確認な部分を考慮して，その伝達関数の弱点を補える制御方法であるところの適応制御の手法を試みた．

本文は，上記の目的を念頭において，その実現の可能性を探るため計算機で簡単なシミュレーションを行い実船実験の可能性を確認した後，実船実験によって実現の可能性を提言するものであり，その具体的な方法と主な結果を簡単にまとめたのもである．もしこの考え方が実船に応用できると，各方向へ船舶移動に対しての微調節が容易になり船舶の安全運行に貢献できるものと思われる．参考にした文献は，最後にまとめてある．

1　船体モデルの設計

試験航海時の汐路丸の排水量が６６０．７tonであることから，船の総重量を６６０．７ton　全長４９．９３m　全幅　１０．０mとする．（文献（１）　参照）

１．１　汐路丸の運動（伝達関数等）について

ここでは，文献（１）を参考に運動力学的に汐路丸の運動を解析してみた．

１．１．１　Ｙａｗ－左旋回について（バウスラスターのみ使用）

Ｙａｗ－右旋回は，潮流の影響を大きく受けていると思われるので，Ｙａｗ－左旋回についてのみ取り扱ってみる．

文献（１）の旋回試験成績から表（１．１）を求め，これを基にグラフ　Ｆｉｇ－１．１　を作る．このグラフから加速度を求めると表（１．２）の様になる．

抵抗係数を求めるため，

運動方程式　$F \times L = I \times \ddot{\theta}(t) + Fr$

（Ｆｒ：抵抗　Ｉ：慣性モーメント　Ｌ：回転の中心と思われる点からスラスターまでの距離　Ｆ：スラスターからの推力）

を考えてみる．

まず表（１．２）をグラフにして（Ｆｉｇ－１．２）$\ddot{\theta}(t)=0$となっている点を求めると，８Step付近が考えられる．

即ち運動方程式は，$F \times L = I \times \ddot{\theta}(t) + Fr = Fr$とすることができる．

さらに船は高速で移動している訳でないので

抵抗$Fr = C \times \dot{\theta}(t)^2$とすることができる．上述した８Step付近では$\dot{\theta}(t)=0.0131$(rad/s)とすることができるので運動方程式は、$I=64927.65$

$L=17.5$m　$F=2.28$ton（この実験で使用したノッチでの平均推力）を代入すると

$$2.28 \times 17.5 = 39.9 = C \times \dot{\theta}(t)^2$$

$$\therefore C = 258224.85$$

又　$F(t) \times L = I \times \ddot{\theta}(t) + C \times \dot{\theta}(t)^2$より

$$\ddot{\theta}(t) = \frac{F(t) - 14755.71 \times \dot{\theta}(t)^2}{3710.15}$$

となる．したがって入力としてバウスラスターからの推力を，出力として角加速度を求める式が得られた．

１．１．２　Ｙａｗ－左旋回（バウ＋スターンスラスター使用）について

“１．１Ｙａｗ－左旋回”と違い，バウスラスター及

びスターンスラスターの両方を使用してみた．（これも右旋回は潮流の影響を大きく受けていると思われるので左旋回のみ見てみた．）

文献（１）の旋回試験成績から，移動した角度　表（１．３）さらに角速度　表（１．４）を求めグラフ　Ｆｉｇ－１．３　とする．これを基に角加速度をグラフ　Ｆｉｇ－１．４　とした．

ここでも抵抗係数を求めるため運動方程式

$$F_B \times l_B + F_S \times l_S = I \times \ddot{\theta}(t) + Fr$$

を考えてみる．ここで　F_B：バウスラスターからの推力でノッチ１０に於て２．２８ton　F_S：スターンスラスターからの推力でノッチ１０に於て１．５６ton　l_B，l_Sは船中心から各々のスラスターまでの距離で各々　１７．５ｍ　１８．８ｍ　Ｆｒ：抵抗　Ｉ：慣性モーメント（Ｉ＝６４９２７．６５）　とした．

Ｆｉｇ－１．３，Ｆｉｇ－１．４　から　$\dot{\theta}(t) = 0.0411$(rad/s)付近で　$\ddot{\theta}(t) = 0$と近似できるので，上述の運動方程式は　$F_B \times l_B + F_S \times l_S = Fr$　とできる．また　Ｆｒは，抵抗で$Fr = C * \dot{\theta}(t)^2$とした．各々運動方程式に代入した結果

抵抗係数　$C = (F_B l_B + F_S l_S) / \dot{\theta}(t)^2$

$= 41182.63$

$$\ddot{\theta}(t) = \frac{0.9F_B(t) + F_S(t) - 2190.57\dot{\theta}(t)^2}{3453.6}$$

となる．この様にして，入力としてバウ，スターン両スラスターからの推力を，出力として角加速度を求める式が求められた．

1．1．3 Baw－Thrusterについて

ここでバウスラスターの動特性について考えるが，その測定方法から船体の回転運動も含んだ動特性であると考えられるので参考程度でしかない．

微分方程式 $\ddot{F}(t)+2\xi\omega_n\dot{F}(t)+\omega_n^2F(t)=\omega_n^2$

を考える．

文献（1）から計って

行き過ぎ量：1．69－1＝

$\exp(-\xi\pi/(1-\xi^2)^{1/2})$

行き過ぎ時間：12(sec)＝

$\pi/(\omega_n(1-\xi^2))$

と仮定すると上記の微分方程式は，

$\ddot{F}(t)+0.078\dot{F}(t)+0.068F(t)$

$=0.068r(t)$

となり，両辺ラプラス変換すると

$s^2F(s)+0.078sF(s)+0.068F(s)$

$=0.068r(s)s^{-1}$

$$\therefore F(s) = \frac{0.068}{s^2 + 0.078s + 0.068} \times r(s)$$

このことから，バウスラスターの動特性に関して（バウスラスターの推力に関しては）は，

入力側		出力側
ノッチ量	$\dfrac{0.068}{s^2 + 0.078s + 0.068}$	推力

とした．

1．1．4　Stern－Thrusterについて

次に、スターンスラスターの動特性について考えるが、“1．3　Baw－Thrusterについて”と同様，その測定方法から船体の回転運動も含んだ動特性と考えられるので参考程度でしかない．

スターンスラスターの伝達関数は，バウスラスターの伝達関数を求めた方法で求め、その結果以下の様になった．

$$\ddot{F}(t) + 0.08\dot{F}(t) + 0.018F(t) = 0.018r(t)$$

両辺ラプラス変換すると，

$$s^2F(t)+0.08sF(s)+0.018F(s)$$
$$=0.018r(s)s^{-1}$$

このことから，スターンスラスターの動特性に関して（推力に関して）以下の様になる．

入力側	$\frac{0.018}{s^2+0.08s+0.018}$	出力側
ノッチ量		推力

1．1．5　横移動について

文献（1）の横移動試験成績から，（バウ，スターン）スラスター各々の　ノッチを（9．1，10．0）にした際のデータを参考に伝達関数を求める．

両スラスターの推力は各々（1．86，1．65）ton　計3．51tonである．

次に，文献（1）を見ると左右に大きな違いがあり（これは，潮流が原因と思われる．）各々の方向で求められないので実験結果を平均して求めてみた．　実験結果（平均値）からグラフ　Fig－1．5　を求め，このグラフから加速度を求めると　Fig－1．6　の様になる．

抵抗係数を求めるため

運動方程式　$F = M \times \ddot{X}(t) + Fr$　を考える．ここでF：スラスターからの推力，M：総重量，Fr：抵抗とした．$\ddot{X}(t) = 0$では，運動方程式は，$F = Fr$　とできる．又抵抗　Frは低速に於いては，摩擦抵抗が８０％以上を占めていると思われるので，他の抵抗は無視して$Fr = C \times \dot{X}(t)^2$とした．

つまり　$3.51 = C \times 0.5^2$となる．

$$\therefore C = 14.04$$

$$\therefore \quad \ddot{X}(t) = \frac{F(t) - 14.04 \times \dot{X}(t)^2}{660.7}$$

以上の様にして，スラスターからの推力と加速度の関係式が求められた．

１．１．６　縦移動について

船舶の運動を次の様に仮定する．$F = \dot{V} \times M + Fr$

（ここでF：推力，M：総重量　V：前方向への速度，Fr：抵抗とした．）

ここでも抵抗は各種考えられるが，“１．５　横移動”と同様に，摩擦抵抗が８０％以上を占めていると思われるので，これ以外の抵抗（空気，渦，造波等）は，無視した．

文献（２）からフルードの式を使って抵抗係数Cは，$C = 16.501$となり運動方程式は以下のようになる．

$$F(t) = \dot{V}(t) \times 660 + C \times V(t)^{1.825}$$

$$\therefore \quad \dot{V}(t) = \frac{F(t) - C \times v(t)^{1.825}}{660.7}$$

1．2　実際のプラントとモデルプラント

汐路丸船体の運動方程式（伝達関数）は，実際の船体運動と多少異なると思われるので，これらを計算機上でモデルプラントの動特性として適する様に微調整して求めた運動方程式（伝達関数）を以下モデルプラントとして実船の運動特性とは異なる物として取り扱う．また汐路丸の実際の厳密な運動方程式は不明であるが，文献（1）にある各々のデータに近づける様に微調整して得られた運動方程式（伝達関数）を以下実プラントとする．

1．2．1　Ｙａｗ－左旋回（バウスラスタのみ使用）について

これは，実プラントとモデルプラントを同じ物を使用した．
つまりモデルプラント，実プラント　：

$$\ddot{\theta}(t) = \frac{F(t) - 61281.52\,\dot{\theta}(t)^2}{32858.06}$$

とした．

1．2．2　Ｙａｗ－左旋回（バウ＋スターンスラスタ使用）について

これは計算機上で微調整を繰り返すことによって，モデルプラントは，

$$\ddot{\theta}(t) = \frac{R - 150 \times \dot{\theta}(t)}{3165.65}$$

Ｒ：目標角速度(rad/s)

実プラントを

$$\ddot{\theta}(t) = \frac{F_S + F_B - 2190.57 \times \dot{\theta}(t)^2}{3165.65}$$

F_S：スターンスラスタからの推力(ton)

F_B：バウスラスタからの推力(ton)

とした．

1．2．3　バウスラスタについて

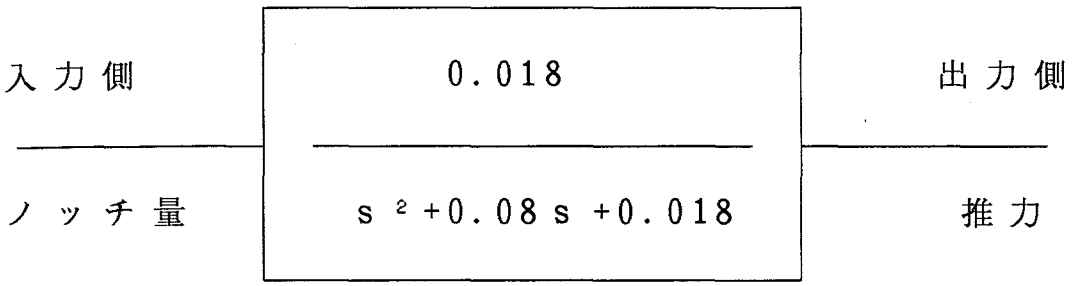
これは，モデルプラントと実プラントを同じ物として

入力側		出力側
	0.068	
ノッチ量	$s^2 + 0.078s + 0.068$	推力

とした.

1．2．4　スターンスラスタについて

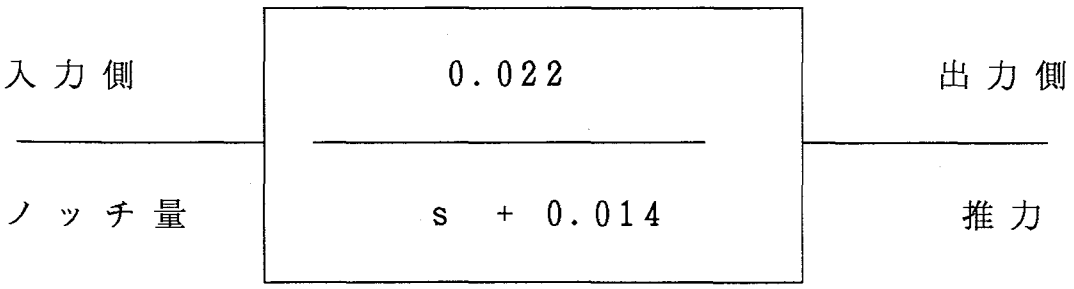
これも，モデルプラントと実プラントを同じ物として



とした.

1．2．5　横移動について

これも計算機上で微調整を繰り返すことにより，モデルプラント：



とした.

モデルプラントは，減衰項が1次であるのに対し，実船は2次であると思われるのでさらに微調整を繰り返すことによって，

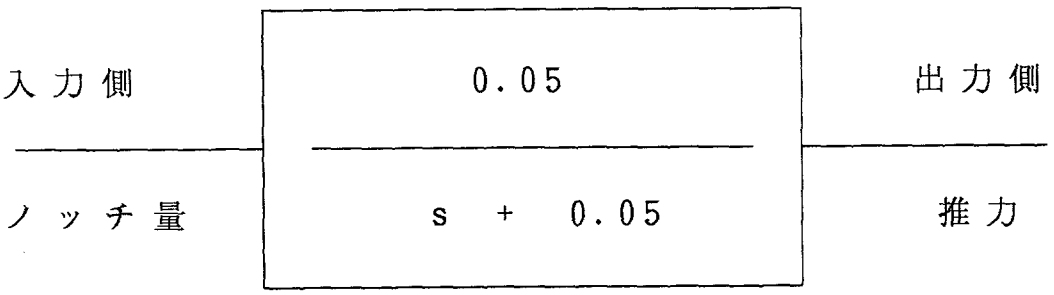
実プラント：$\ddot{X}(t) + 0.012\,\dot{X}(t)^2 = 0.005$

とした．

1．2．6　縦移動について

これも計算機上で微調整を繰り返すことにより，

モデルプラント：



として，実プラント：

$$\ddot{y}(t) = \frac{F(t) - C\,\dot{y}(t)}{660.7}$$

とした．

（C：前出の抵抗係数　F：プロペラからの推力）

＊　注意　＊

文献（１）にあるデータ上で
横移動については，Ｘ方向
縦移動については，ｙ方向　その速度を，Ｖ
とした．
旋回については，　船首の左回転をθとした．

deg	角度差 θ (deg)	角速度 $\dot{\theta}$ (rad/s)
0.0		
	10.0	0.022
10.0		
	11.0	0.024
21.0		
	14.0	0.031
35.0		
	5.0	0.011
40.0		
	6.0	0.013
46.0		
	8.0	0.013
54.0		
	6.0	0.013
60.0		
	6.0	0.013
66.0		
	6.0	0.013
72.0		
	6.0	0.013
78.0		
	10.0	0.022
88.0		
	10.0	0.022
98.0		
	2.0	0.0044
100.0		
	4.0	0.0087
104.0		
	5.0	0.011
109.0		
	4.0	0.0087
113.0		
	3.0	0.0065
116.0		
	2.0	0.250
118.0		

表（1．1）

step	角加速度(rad/s)
0～ 0.5	0.0028
0.5～ 1.0	0.0003
1.0～ 1.5	0.0003
1.5～ 2.0	0.0009
2.0～ 2.5	0.0009
2.5～ 3.0	−0.0025
3.0～ 3.5	−0.0003
3.5～ 4.0	0.0003
4.0～ 4.5	0.0025
4.5～ 5.0	0.0
5.0～ 5.5	0.0
5.5～ 6.0	0.0
6.0～ 6.5	0.0
6.5～ 7.0	0.0
7.0～ 7.5	0.0
7.5～ 8.0	0.0
8.0～ 8.5	0.0
8.5～ 9.0	0.0
9.0～ 9.5	0.0
9.5～10.0	0.0011
10.0～10.5	0.0011
10.5～11.0	0.0
11.0～11.5	0.0
11.5～12.0	−0.0022
12.0～12.5	−0.0022
12.5～13.0	0.0006
13.0～13.5	0.0005
13.5～14.0	−0.0003
14.0～14.5	−0.0003
14.5～15.0	−0.0003
15.0～15.5	−0.0003
15.5～16.0	0.0003
16.0～16.5	0.0003
16.5～17.0	−0.0003
17.0～17.5	−0.0003

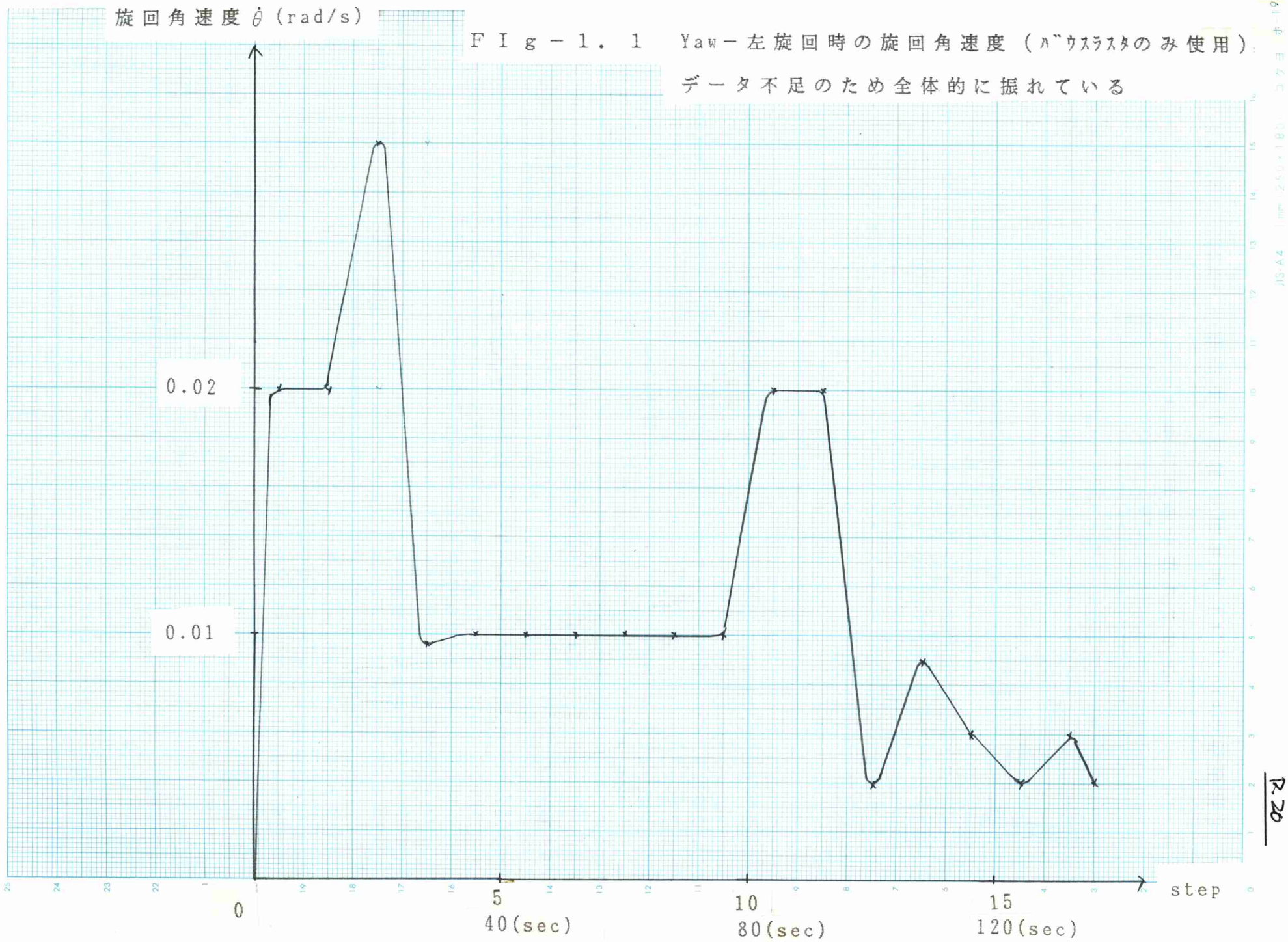
表（1．2）

step	角度(rad)
0	0
1	0.05
2	0.23
3	0.51
4	0.84
5	1.13
6	1.57
7	1.90
8	2.16
9	2.58
10	2.79
11	3.14

表（1.3）

step	角速度(rad/s)
0	
	0．0063
1	
	0．0225
2	
	0．0350
3	
	0．0410
4	
	0．0360
5	
	0．0550
6	
	0．0410
7	
	0．0330
8	
	0．0530
9	
	0．0260
10	
	0．0440
11	

表（1．4）



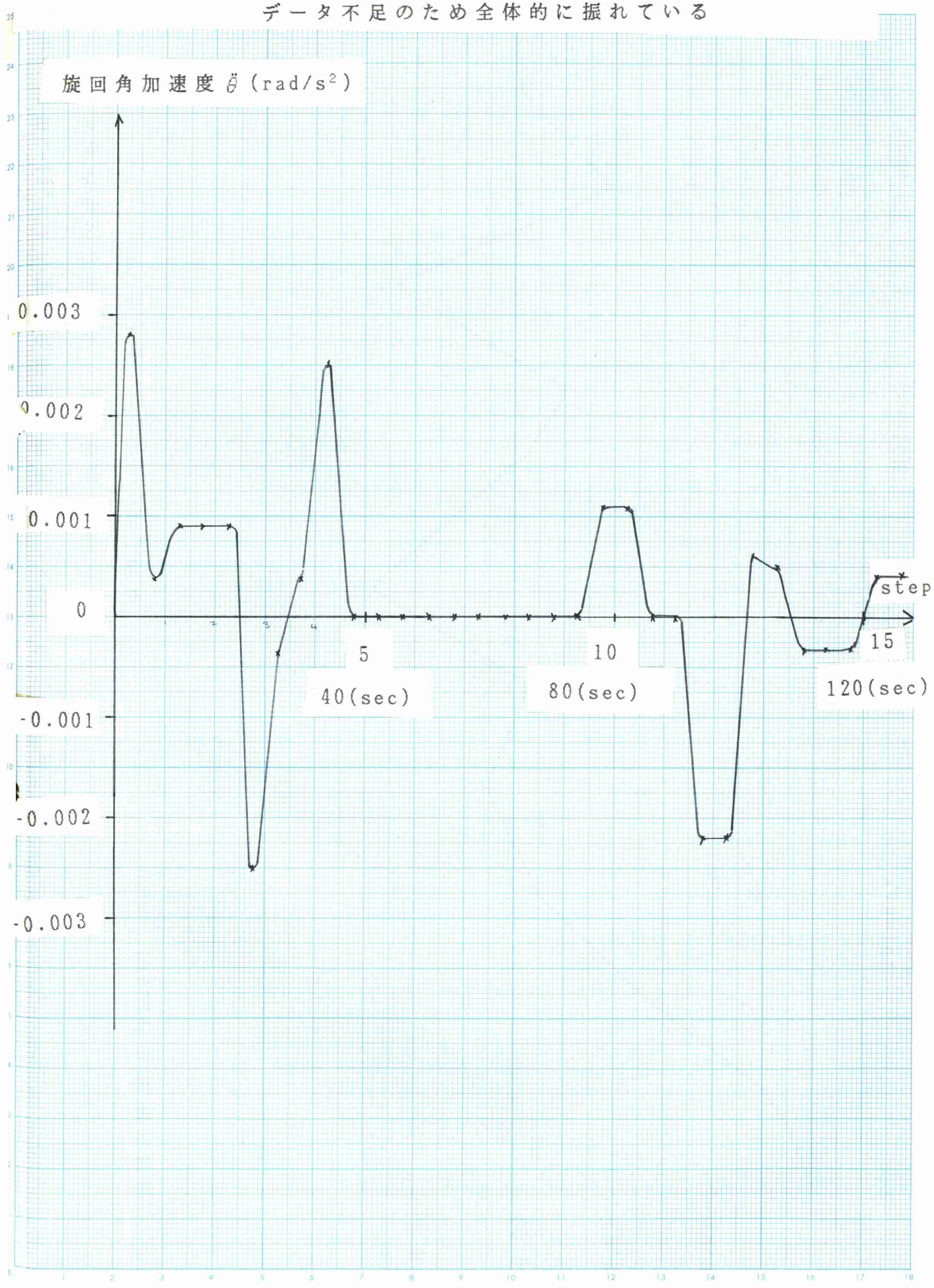
F I g − 1 . 1　Yaw − 左旋回時の旋回角速度（ﾊﾞｳｽﾗｽﾀのみ使用）

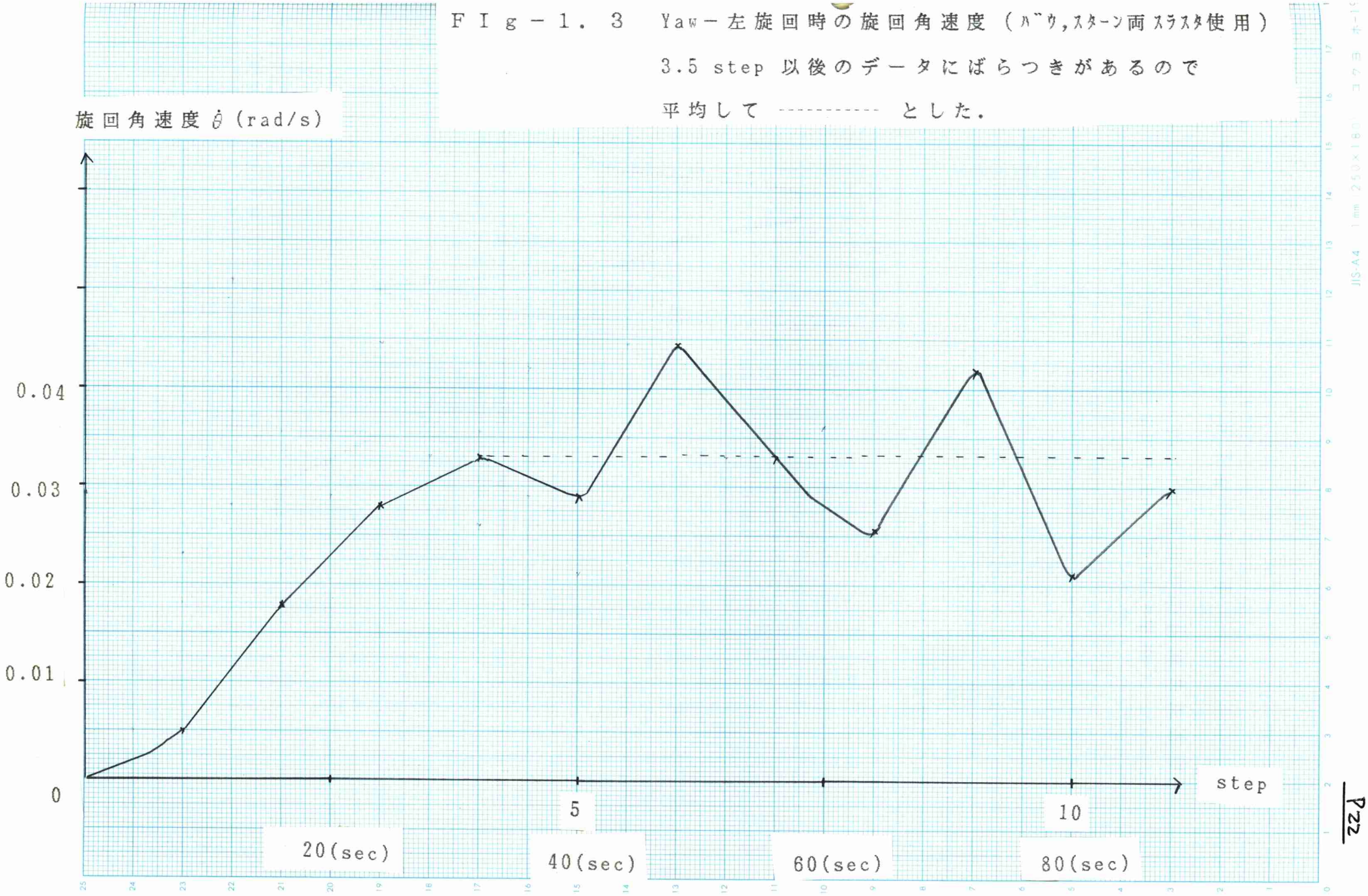
データ不足のため全体的に振れている

P.20

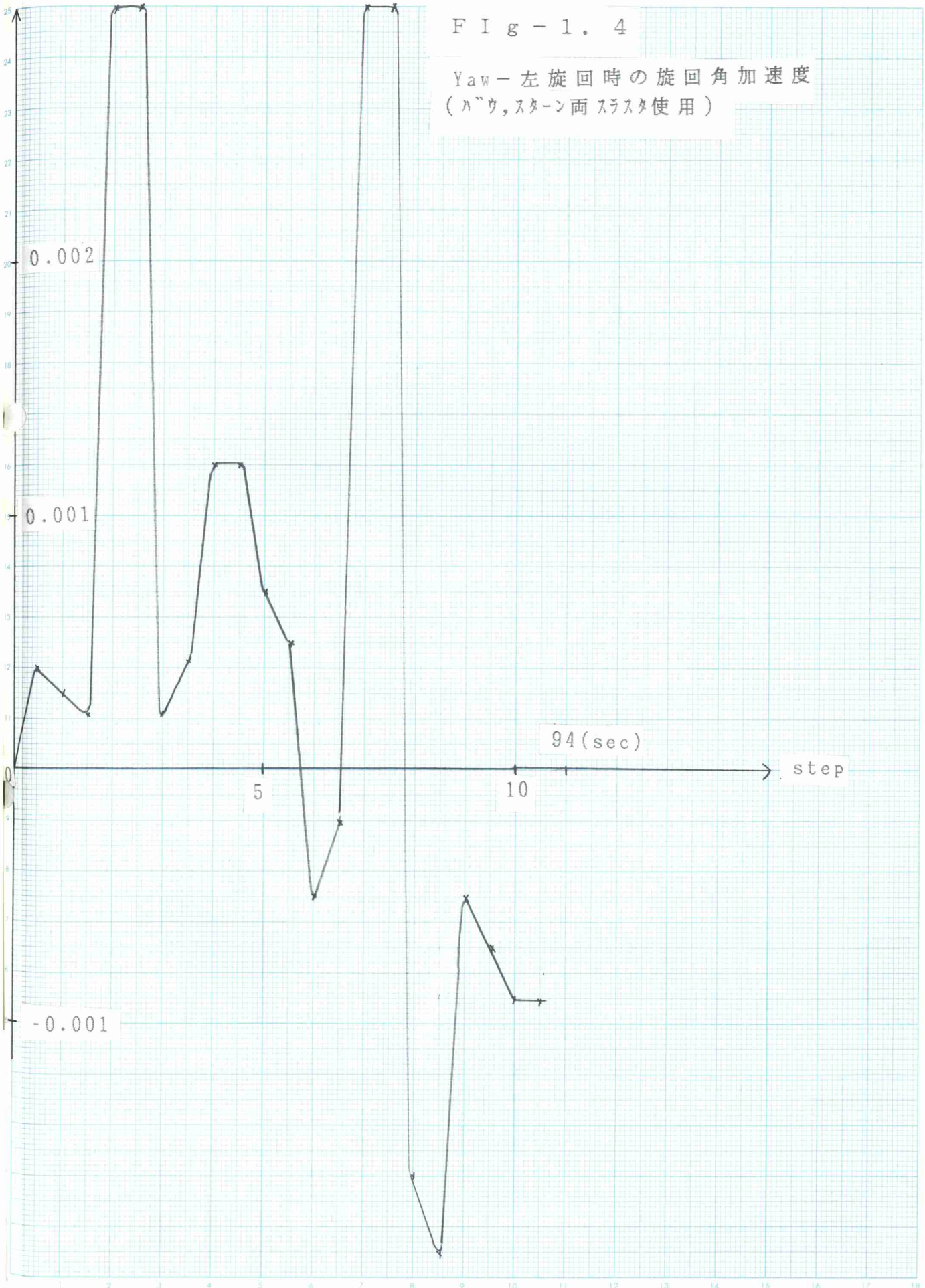
Ｆ Ｉ ｇ － １． ２　　Yaw－左旋回時の旋回角加速度（ﾊﾞｳｽﾗｽﾀのみ使用）

データ不足のため全体的に振れている



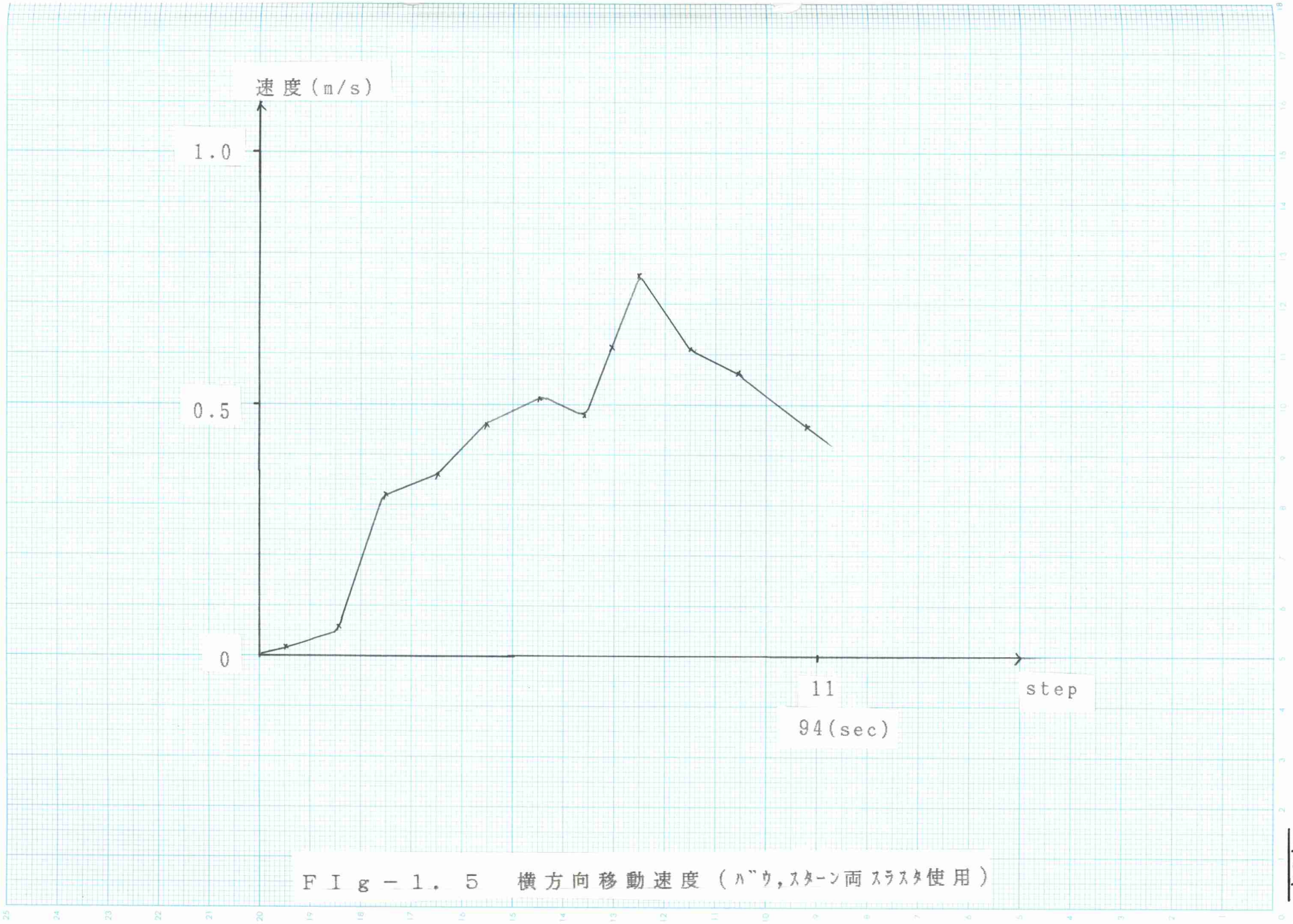

Ｆｉｇ－１．３　Yaw－左旋回時の旋回角速度（ﾊﾞｳ,ｽﾀｰﾝ両ｽﾗｽﾀ使用）
3.5 step 以後のデータにばらつきがあるので
平均して ---------- とした．
旋回角速度 $\dot{\theta}$ (rad/s)
0.04
0.03
0.02
0.01
0
step
5
10
20(sec)
40(sec)
60(sec)
80(sec)
Pzz

旋回角加速度 $\ddot{\theta}$ (rad/s²)

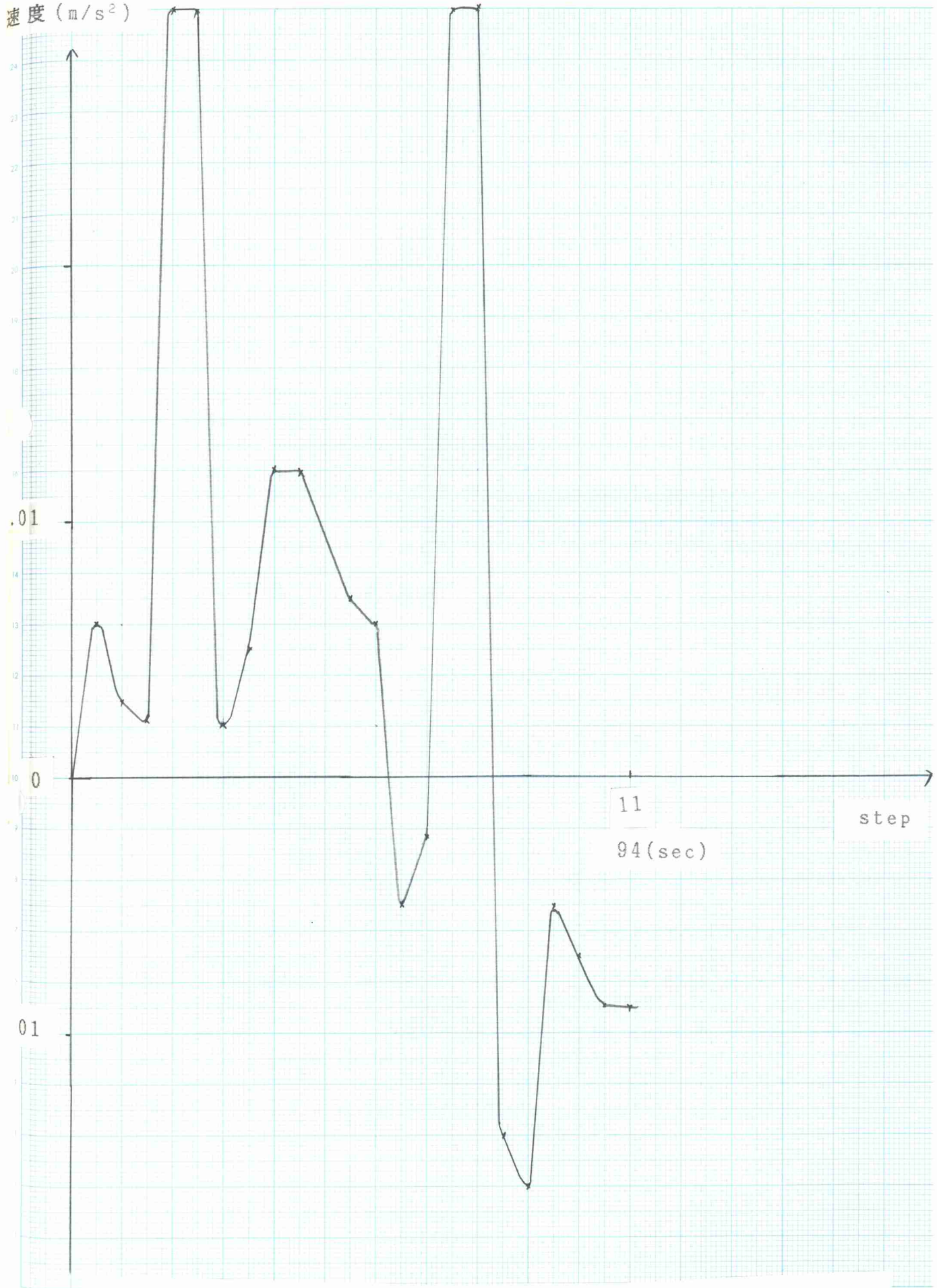


Fig－1．4

Yaw－左旋回時の旋回角加速度
(ﾊﾞｳ,ｽﾀｰﾝ両ｽﾗｽﾀ使用)



FIg－1．5　横方向移動速度（ﾊﾞｳ,ｽﾀｰﾝ両ｽﾗｽﾀ使用）



Ｆ Ｉ ｇ － １． ６　横方向移動加速度（ﾊﾞｳ,ｽﾀｰﾝ両ｽﾗｽﾀ使用）

2　制御アルゴリズム

2．1　適応制御の特徴

前述（始めに）の様に，適応的手法によって船舶の運動制御を試みるが，その中でも今回は，モデル規範型適応制御（ＭＲＡＣ）システムと呼ばれるものを考えてみた．このモデル規範型適応制御システムとは，プラント出力が基準となるモデルの動作に追従する様に，プラントに対する適応アルゴリズムを構成し通常のフィードバックゲイン制御より，高度な制御成績を達成させようと言うものである．その特徴としては，

その１）　大きく外乱を受けるシステムや制御中に大きなパラメータ変動を生じるシステムの制御に用いると，それらの外乱及びパラメータ変動による制御成績への悪影響は排除され，プラント出力がモデルの理想的な出力に追従してくれる．

その２）　プラントの未知パラメータを通常のフィードバック制御の様に様々な方法を測定する必要が無い．つまりプラントのパラメータ測定と制御を同時に行える．

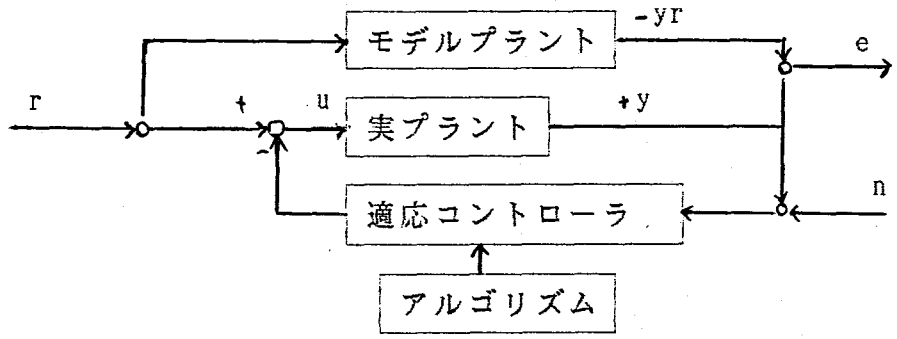
等と言う事が上げられる．

この考え方は，１９５０年代末に登場し，１９８０年代までに理想的な場合の理論的基礎は，ほぼ確立した．

２．２　適応制御アルゴリズムの紹介

Ｆｉｇ２－１のＭＲＡＣシステムを考える．

ここで適応コントローラは，$u = \hat{\theta}^T(t)Z$ とするが，上図のシステムを構成するに際し，$\hat{\theta}$ を生成する適応アルゴリズムを選択しなければならない．そこで典型的な（１）のアルゴリズムを考えてみる．



u ：制御入力　$u = \hat{\theta}^T(t)Z$

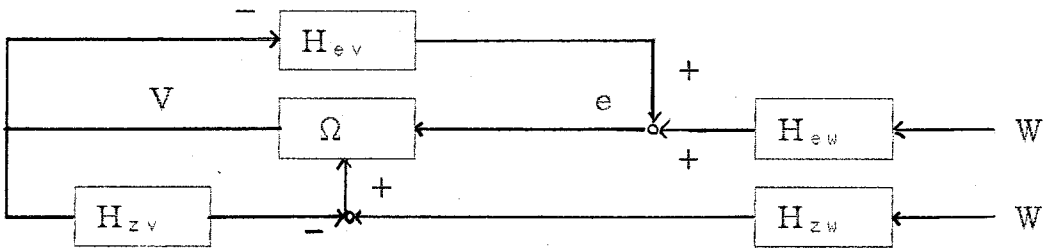
$\hat{\theta}^T$：適応ゲインベクトル　$\hat{\theta}^T = [\theta_1, \theta_2]$

Ｆｉｇ２－１　ＭＲＡＣ_system の基本構成

$$\frac{d\hat{\theta}}{dt} = \Gamma Z e \quad \text{——} \quad (1)$$

Z : Regressor signal　$Z^T = [-y \quad , \quad r]$

この（Ｆｉｇ２－１）ＭＲＡＣシステムに対応するAdaptive error system は，下図（Ｆｉｇ２－２）の様になる．



V：adaptive control error　$V = Z^{T}\theta$

w：目標値ｒ　外乱ｄ　観測ノイズｎ　からなる外部信号

$w(t) = (r^{T}, d^{T}, n^{T})^{T}$

Ｆｉｇ２－２　Adaptive error system

まず adaptive control error を次のように定義する．$\theta(t) = \hat{\theta}(t) - \theta^{*}$　ここでθ^{*}は，Tuned gain（制御が完了した時点で想像される適応ゲイン）と呼ばれる定ベクトルである．

この式から，適応制御信号　$V = Z^{T}\theta$　を使い，

$u = Z^{T}\theta^{*} - V$　となり

Interconnection system　は，次の様になる．

$$\begin{bmatrix} e \\ Z \end{bmatrix} = \begin{bmatrix} H_{ew} & -H_{ev} \\ H_{zw} & -H_{zv} \end{bmatrix} \begin{bmatrix} W \\ V \end{bmatrix} \quad \text{——} \quad (2)$$

上式の構造の特徴は，adaptive control error Vをシステムの残りの部分から切り離すところにある．適応制御が完了すれば，$\theta = 0$ かつ $V = 0$ となる．この場合の Tuned regressor signal は，各々 $e = H_{ew}W$, $Z = H_{zw}W$ となる．つまり adaptive error system の型では，次の様になる．

$$\theta(t) = \hat{\theta}(t) - \theta^{*}(t)$$

$$u(t) = Z^{T}\theta + V$$

$$e = e^{*} - H_{ev}V$$

$$Z = Z^{*} - H_{zv}V$$

ＭＲＡＣシステムが大域的安定となるためには，$H_{ev} \in$ ＳＰＲ（強制実）でなければならない．しかしながら，式（１）のアルゴリズムでは，動的寄生要素の存在によって，この条件が満たされない場合もあるのでロバストなアルゴリズムであるところの，次式を採用することにした．

$$\frac{d\hat{\theta}}{dt} = -\Gamma Z D Z^{T} + \Gamma Z e \quad \text{——} \quad (3)$$

ここで，D(S)は，適応スピードを上げた際にシステムが不安定になることがあるので，よりシステムの安定性を確保するための伝達関数であり，Γは適応スピードを決めるパラメータある．ＭＲＡＣシステムが大域的安定

となる条件は，H_{zu}がstableかつ$H_D \in SPR$でる．

ただし，$H_D = H_{ev} - H_{zv} D = H_{evD} + D$

$$H_{evD} = [\,1 - D(S)\theta_1\,]\, H_{ed}(S)$$

からなる伝達関数である．

ロバスト安定性を補償する伝達関数D(S)は，

$$D(S) = D_0 \frac{TS}{TS+1} \qquad (D_0 > 0,\ T > 0)$$

とし，H_DがSPR条件を満たす様にD_0とT（ナイキスト線図，ボード線図等の用いて）を設計する．（文献3，文献5）

3　適応制御によるシミュレーション

3．1　補償器無しの場合

シミュレーションの方法は，”1　汐路丸の運動”で求めたモデルプラントを”2　制御アルゴリズム”で述べたブロック線図中の規範モデルに当てはめ，同じく実プラントをプラントに当てはめ、”2　制御アルゴリズム”で述べた適応アルゴリズム（詳しくは，例えば文献（3）を参照）を用いて制御を行った．

ここでは，もちろんバウ，スターン両スラスターの最大推力が文献（1）にある最大推力を越えてないことを確認した．以下にその代表例を示し簡単な説明を加えるが，シミュレーションに使った適応スピードは内部信号に違いがありフィードバック側，フォワード側各々の数値の上がり方を合わせる為に違いを持たせてみた．

3．1．1　横移動（速度制御）について

このシミュレーションは，離着岸時に必要な横移動のうち速度を適応制御を用いて制御したものであり，参考までにスラスターからの推力の合計及び加速度の変化も見ることにした．

図例は、ステップ入力（目標値として横方向への速度

０．４５ｍ／ｓ　約１ノット）を入れてシミュレーションを行った物である．ここでの適応スピードは、フィード（バック，フォワード）各々（１，０．０２）で行ってみた．

またシミュレーションの途中で制御が成されたと思われる時点で，目標値をいったん速度０ｍ／ｓとして，再び目標値を元に戻す事を繰り返す．　（３００ｓｅｃ毎）

この図例からモデルプラントの理想的な変化に対し計算機上の実プラントは適応ゲイン，初期値共に０から始めている為立ち上がりこそ遅れているが，プラント出力である所の船体のスピードが理想的なモデルプラントの出力に追従し良好な制御を行っていることが判る．また加速度は，モデルプラントのそれに比べはるかに小さく図中に出て来なかった．（Ｆｉｇ－３．１参照）

３．１．２　Ｙａｗ－左旋回（角速度制御）について

次に旋回時の角速度制御の簡単なシュミレーションを行ってみた．ここではバウ，スターン両スラスターを用いたものと仮定して行っている．

これも前節と同様各々のスラスターの推力，角加速度も見てみた．又適応スピードは，フィード（バック，フォワード）各々（４００，４０）としてみた．

図例は、目標値１．５（ｄｅｇ／ｓ）で左旋回を試み

たものであり，ここでも制御が成されたと思われる時点で目標値を０（ｄｅｇ／ｓ）として，再び目標値を元に戻すことを繰り返した．（４００ｓｅｃ毎）

この図例からもモデルプラントの理想的な変化に対し計算機上の実プラントは良好な制御を行っていることが判る．（Ｆｉｇ－３．２参照）

３．１．３　縦方向（速度制御）について

さらに船の前進（後進）時における速度制御も簡単なシュミレーションを行ってみた．（注：これは、あくまで参考程度に行った物である．）ここでは適応スピードは，フィード（バック，フォワード）各々（４０，２）とした．

図例は，目標値０．４５（ｍ／ｓ）で前進を試み，ここでも制御が成されたと思われる時点で一旦目標値を０（ｍ／ｓ）として，再び目標値を元に戻す事を繰り返した．（３００ｓｅｃ毎）

この図例は，あくまでも参考として行った物であるが，これもやはり良好の制御を行っている．
（Ｆｉｇ－３．３　参照）

＊　注意　＊
プロペラのスラスト最大７tonと仮定した．

3．2　補償器を使用した場合

前節では，補償器は使わずシミュレーションを行ってみたが，ここでは補償器を使用し，その有効性を確認した．補償器の設計法と有効性，及び適応スピードとの関係はすでに文献（3）で述べられている．

3．2．1　横移動（速度制御）について

図例は，”3．1．1　横移動について”でのシミュレーションと同じ条件で適応スピードを各々2倍にしてみたものである．またD_0＝0．3　T＝1000とした．

この図例は動的寄生要素にバウ，スターン両スラスタの動特性を当てはめてみた物であり，寄生要素としてはかなり大きいと思われるが，適応スピードを2倍にしても速度は不安定になっていない．（Fig－3．4参照）

3．2．2　Yaw－左旋回（角速度制御）について

図例は，”3．1．2　Yaw－左旋回”でのシミュレーションと同じ条件で，適応スピードを4倍にしてみた物である．またD_0＝0．1　T＝100とした．この図例でも動的寄生要素にバウ，スターン両スラスタの動特性をあてはめてみた．ここでは適応スピードを4倍に上げてみたが，良好な制御を行っている．

（Ｆｉｇ－３．５参照）

３．２．３　縦移動（速度制御）について

図例は，”３．１．３　縦移動について”でのシミュレーションと同じ条件で，適応スピードを６倍にしてみた物である．またＤ$_0$＝０．２　Ｔ＝１とした．　ここでは，プロペラの動特性が不明であるため動特性を動的寄生要素とすることができず，以下の物を動的寄生要素と仮定して行ってみたが，良好な制御を行っているのが判る．（Ｆｉｇ－３．６参照）

動的寄生要素：$\dfrac{1}{S+1}$

＊　最後に今回のシュミレーションに使用した主な定数について簡単にまとめる。（計算機上のプラント）

m	船の総重量	６６０．７ton
I	慣性モーメント	６４９２７．６５
C	抵抗係数（回転）	４１１８２．６３
	抵抗係数（前後進）	１６．５０１

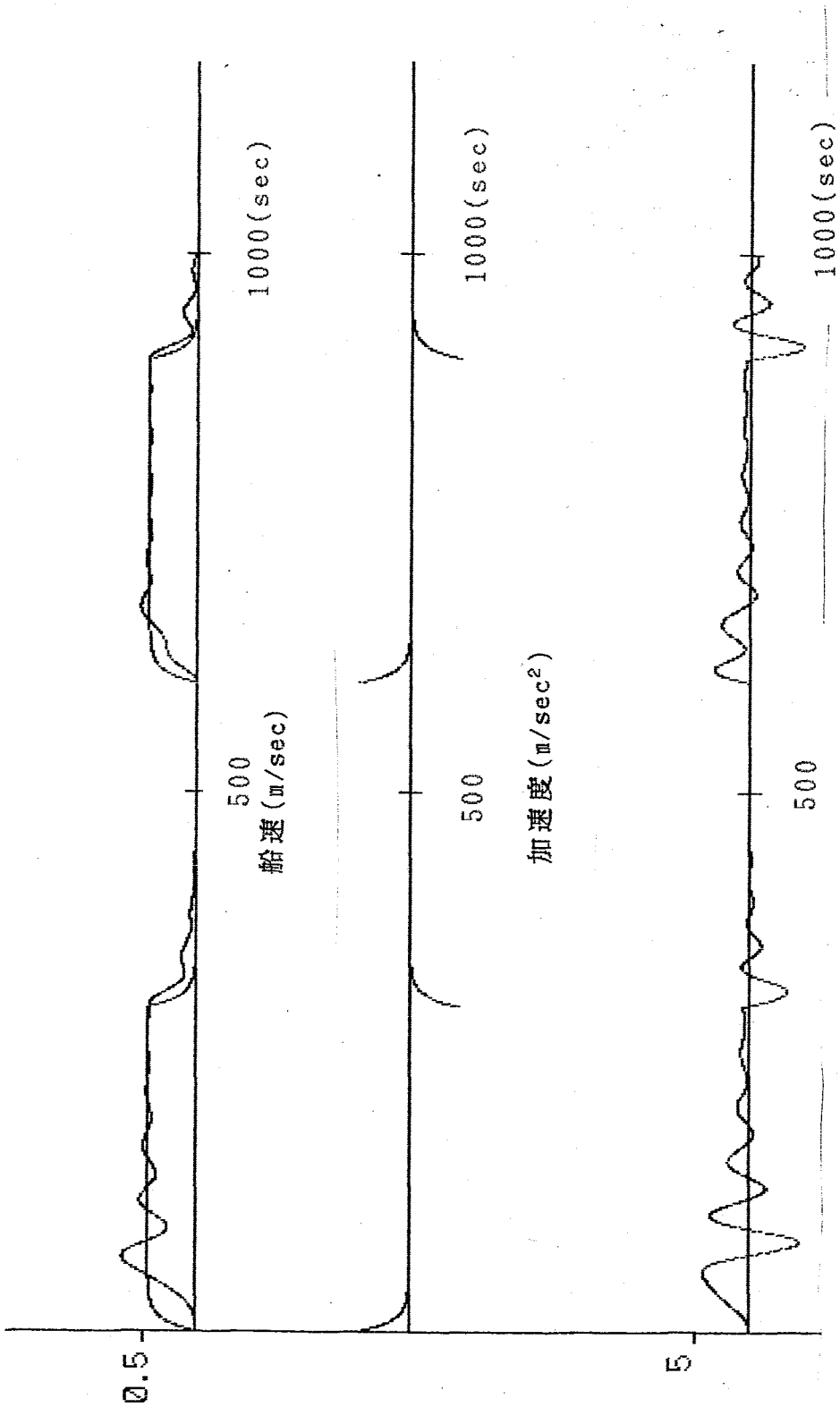

0.5
船速(m/sec)
500
1000(sec)
加速度(m/sec²)
500
1000(sec)
5
推力(ton)
500
1000(sec)

Fig-3.1

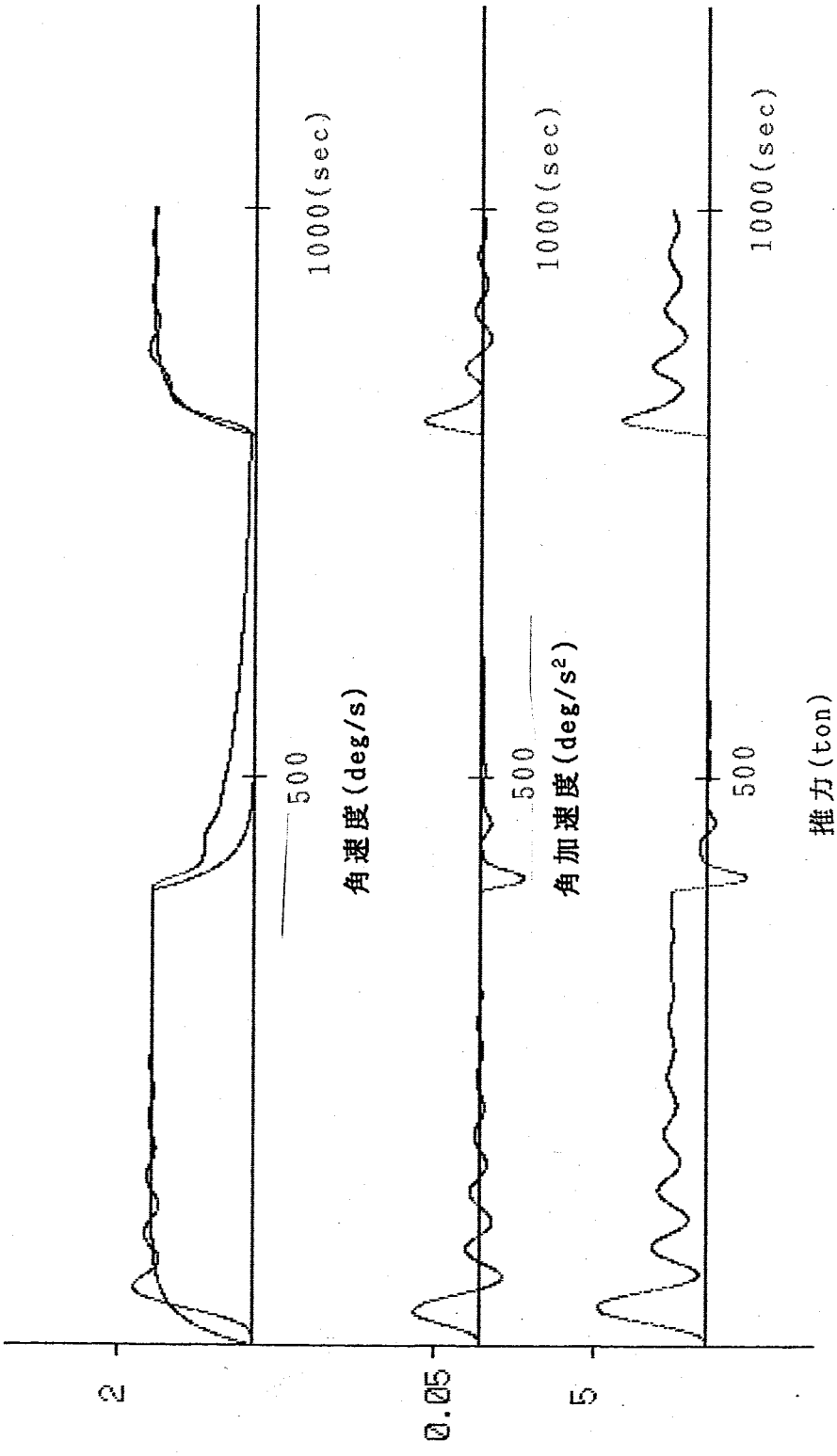
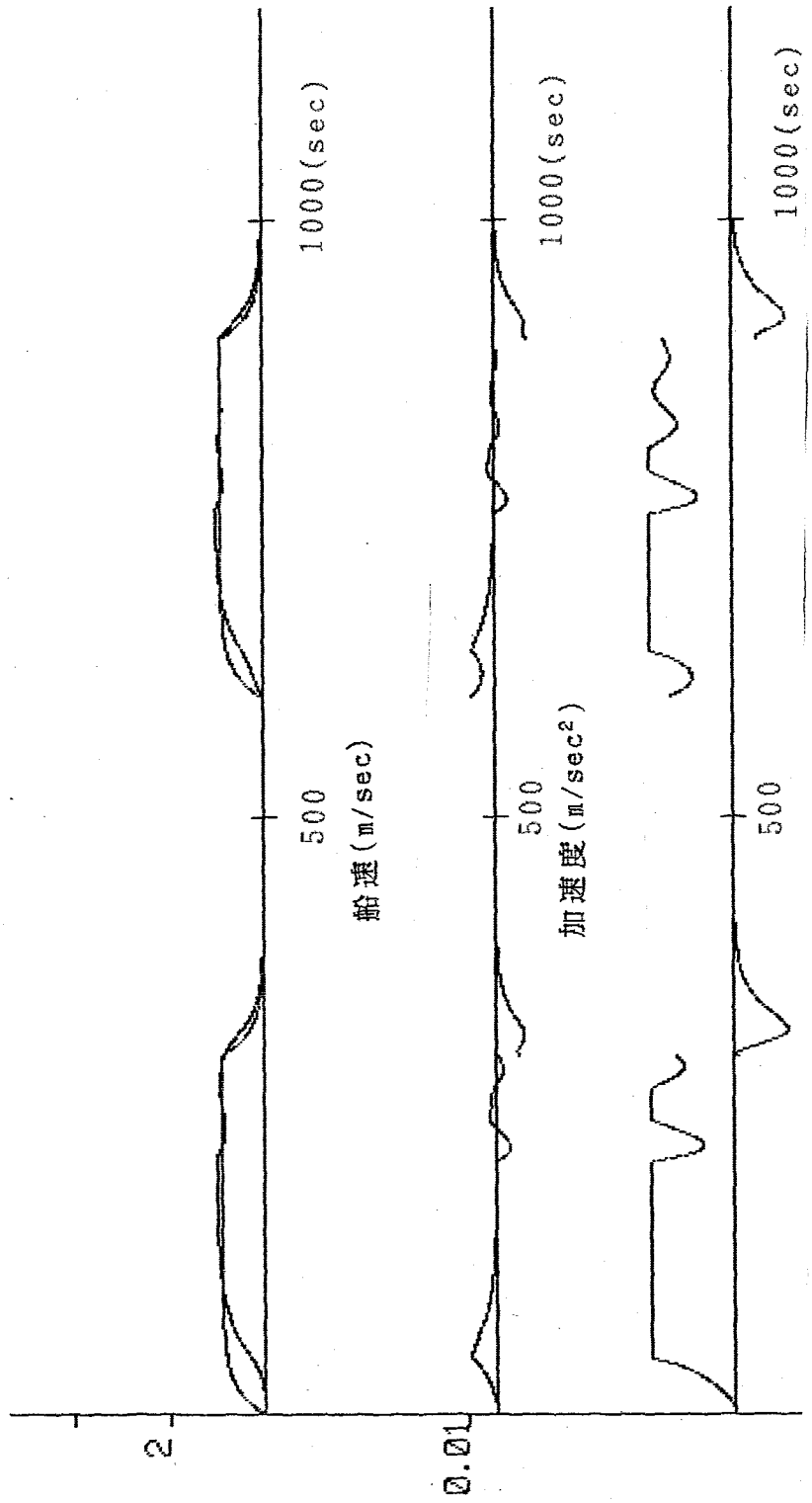
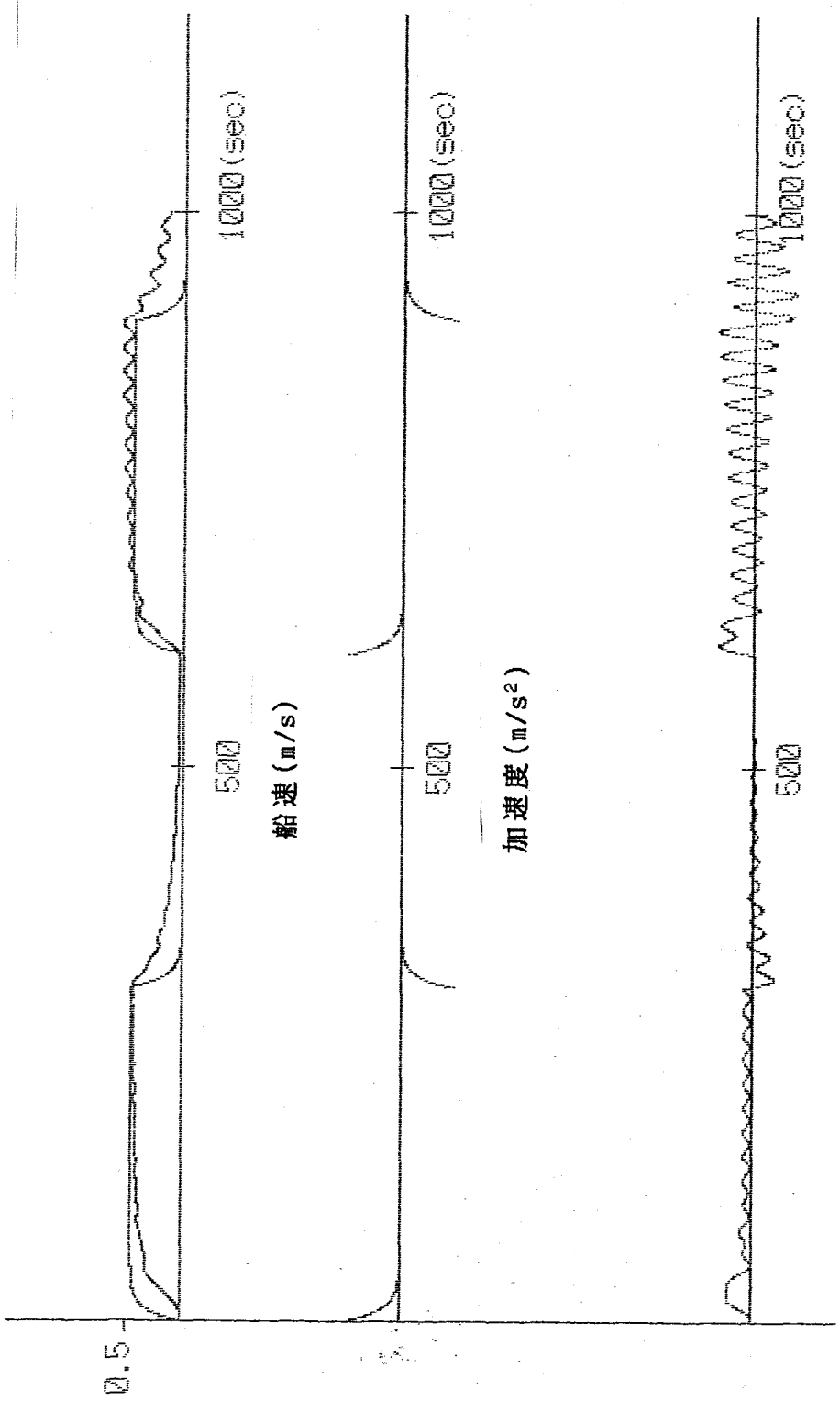

2
0.05
5
1000(sec)
500
角速度(deg/s)
1000(sec)
500
角加速度(deg/s²)
1000(sec)
500
推力(ton)

Fig-3.2


2
0.01
500
1000(sec)
船速(m/sec)
500
1000(sec)
加速度(m/sec²)
500
1000(sec)
推力(ton)

Fig－3．3



Fig－3．4

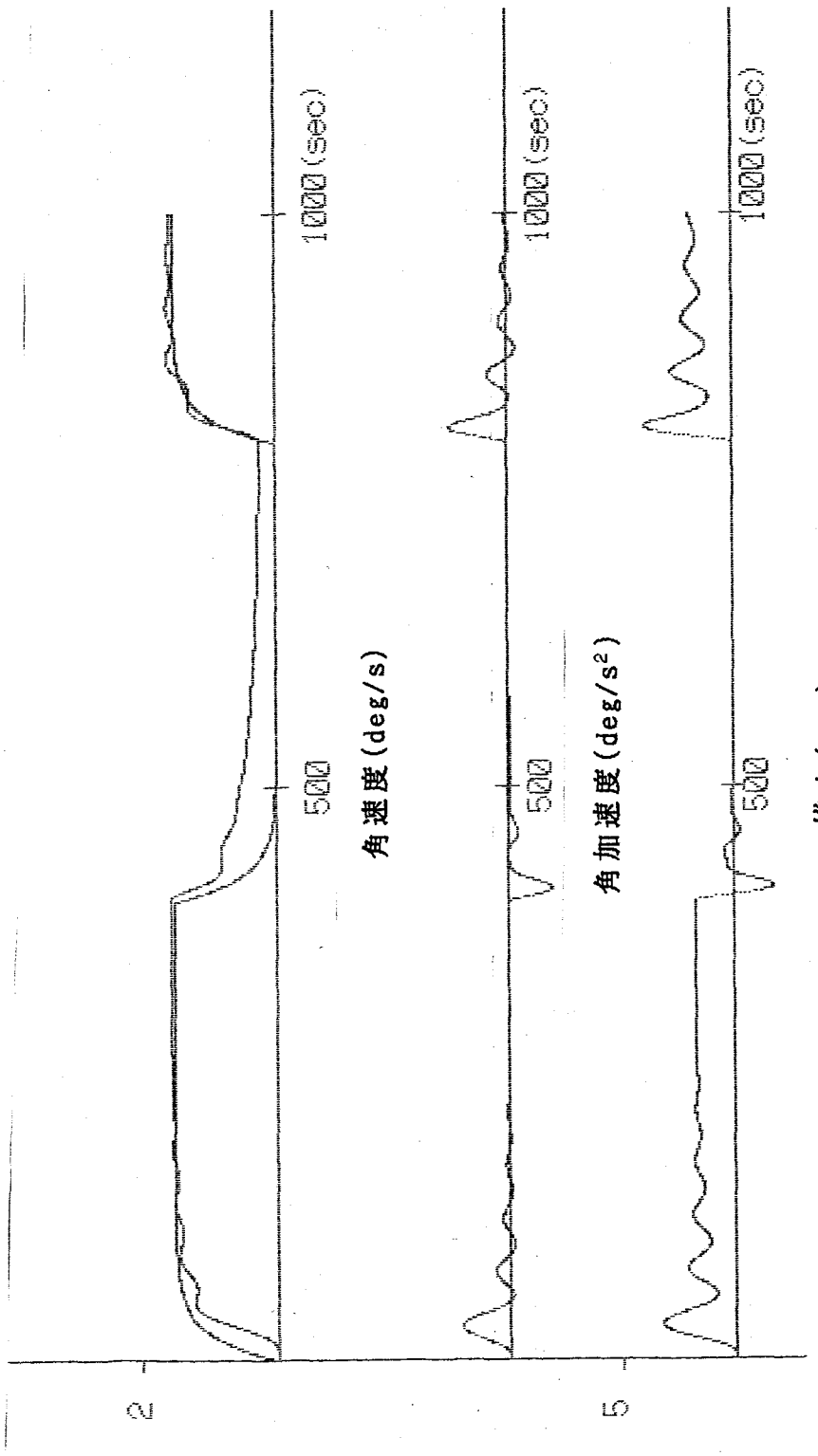

1000(sec)
500
角速度(deg/s)
1000(sec)
500
角加速度(deg/s²)
1000(sec)
500
推力(ton)
2
5

Fig-3.5

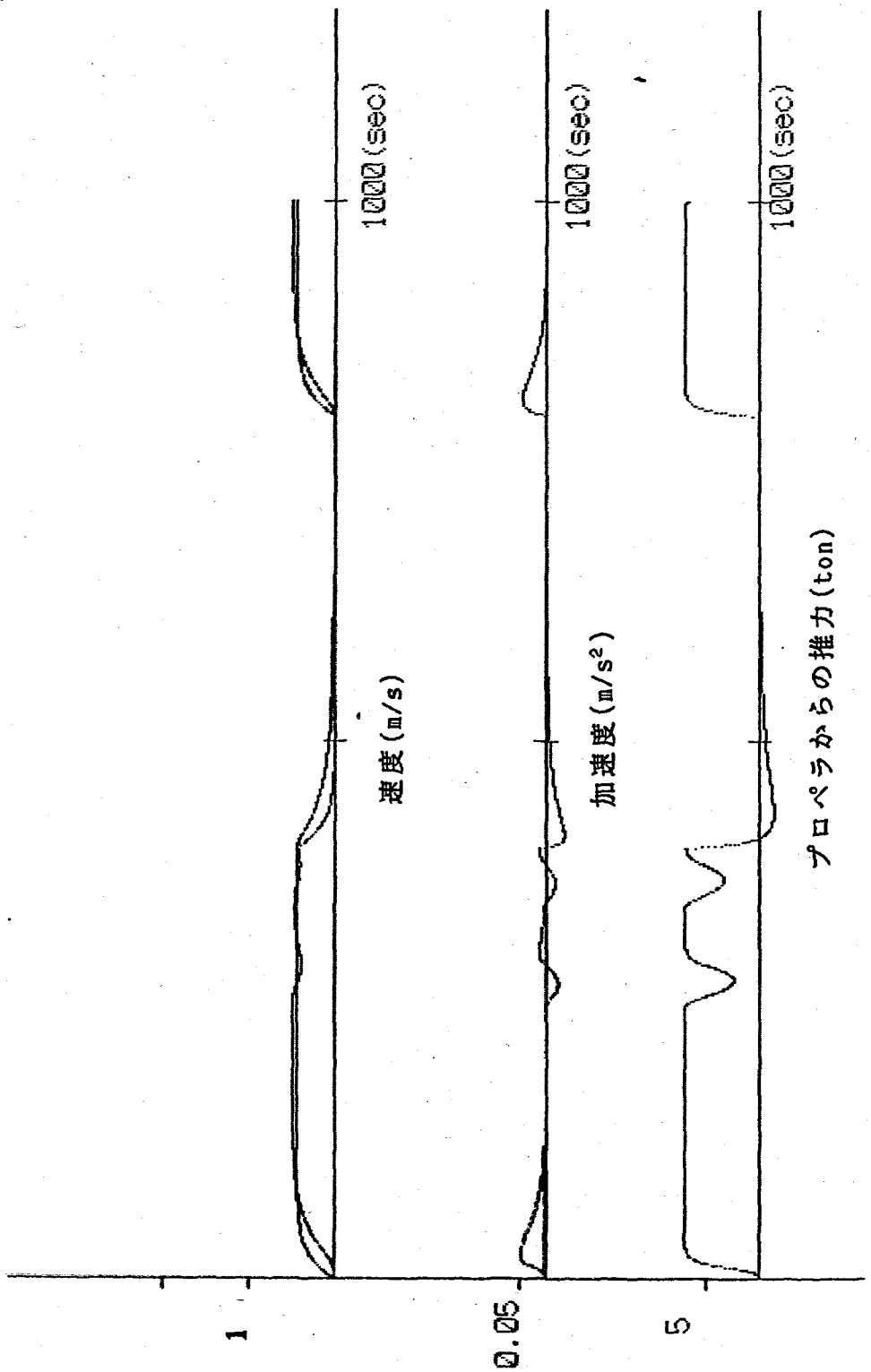


Fig-3.6

4　実船実験

”3　適応制御のシミュレーション”の結果から，適応的手法による船舶の運動制御が，実船実験可能であると思われたので東京商船大学練習船汐路丸（汐路丸主要項目参照）を使用して実船実験を試みた．

汐路丸主要項目

船型	全通船楼甲板型	航行区域	近海区域
全長	４９．９３ｍ	垂線区間	４６．００ｍ
型幅	１０．００ｍ	型深さ	３．８０ｍ
型喫水（満載時）	３．０１ｍ	排水量（満載時）	７１７．５２ｔｏｎ
総トン数	４２５ｔｏｎ		

主機関　ダイハツディーゼル６－ＤＬＭ－２６ＳＬ

４サイクル中速ディーゼル機関

連続最大出力×回転数	１４００ｐｓ×７００ｒｐｍ
最大スラスト	１０ｔｏｎ
プロペラ	かもめ４翼可変ピッチハイスキュー型
直径×回転数	２２００ｍｍφ×３００ｒｐｍ

スターンスラスタ　住吉重工　ｔ－５０型ウオータジェット

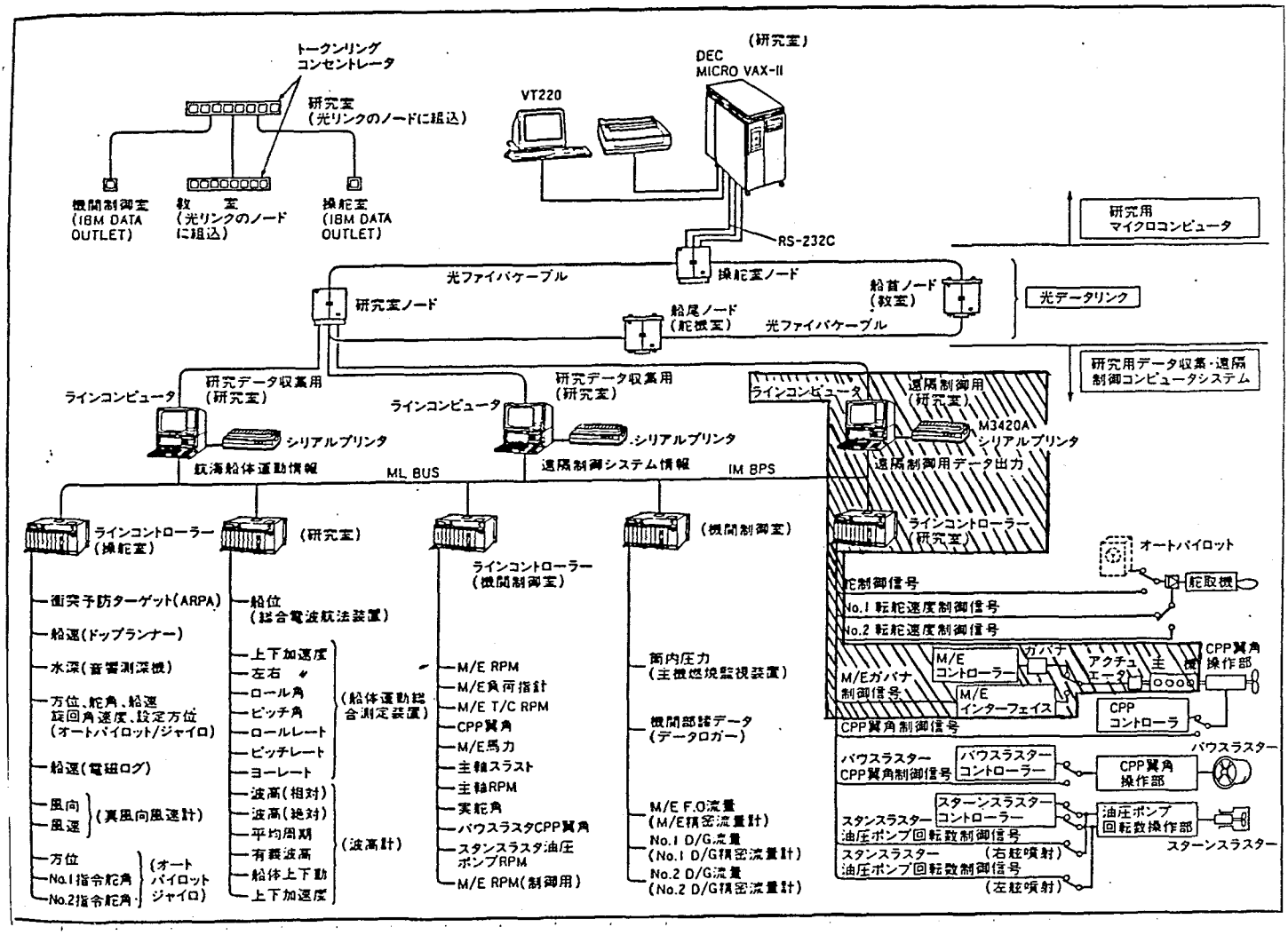
最高回転数	１０７０ｒｐｍ
最高回転時	１８６３ｋｇ

バウスラスタ　ＴＣ－８５Ｎ

入力馬力	１５０ｋｗ
入力回転数	１８００ｒｐｍ
公称スラスト	２４００ｋｇ

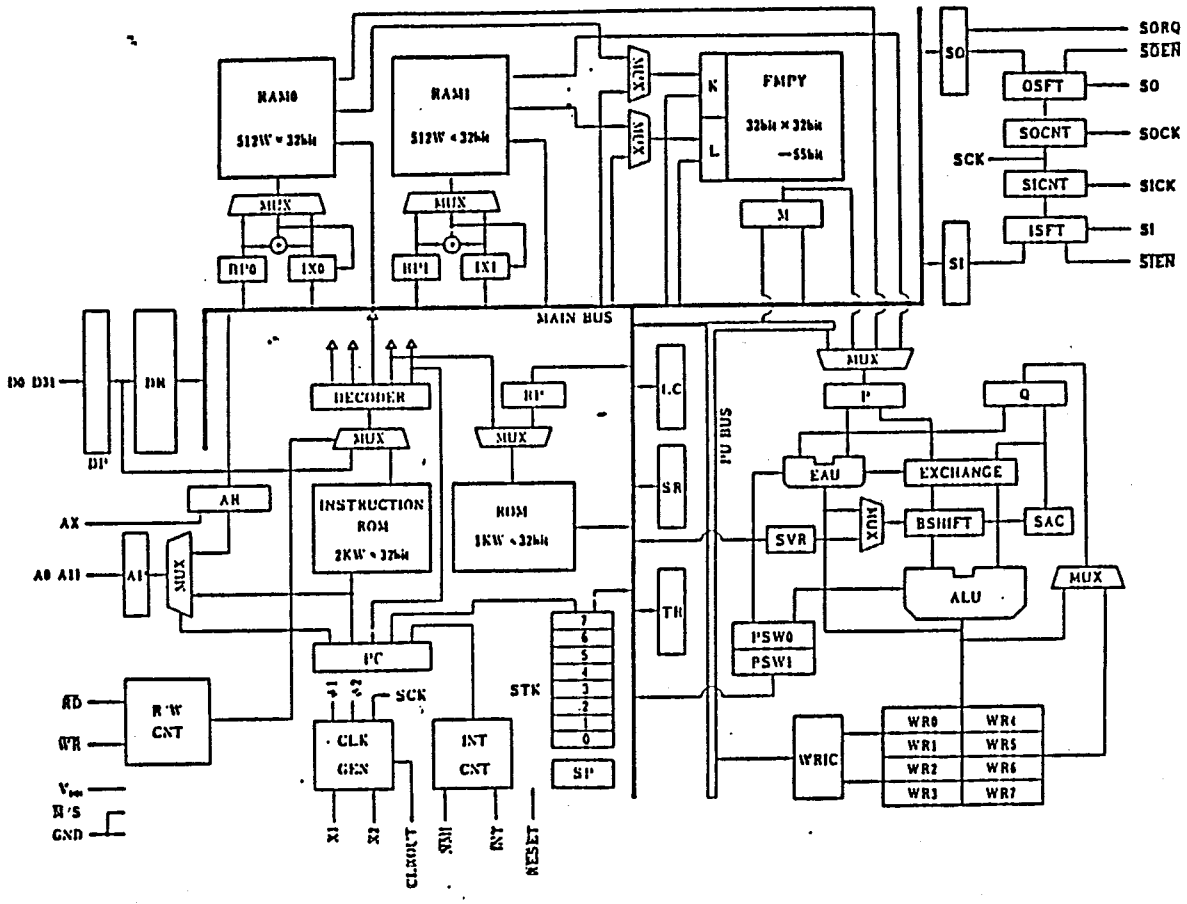
4．1　実験準備

汐路丸には，以下の図に示す船内ＬＡＮシステムと計算機が備わっている．



Ｆｉｇ４－１　汐路丸の船内ＬＡＮシステム

適応制御則は，大量の計算量が必要とされる．そこで今回は，船内ＬＡＮ中のラインコンピュータを使用せずに，デジタルシグナルプロッセサ（ＤＳＰ）を使用した．使用したＤＳＰは，ＮＥＣ社製μＰＤ７７２３０で，メモリーと共に拡張ボードとしたものを利用した．（マイテック社製ＭＳＰ－７７２３０）　以下にμＰＤ７７２３０のブロック線図を示す．



Ｆｉｇ４－２　ＤＳＰのブロック線図

＊ＤＳＰ使用例

今回使用したＤＳＰの使用例をプログラム構造と共に示す．（プログラムは，"a"ドライブにあると仮定する）

ｍａｉｎ１．ｓｒｃ　：メインプログラム

ｍａｉｎｓ０．ｓｒｃ：適応制御を行う時，必要とおもわれるプログラム

ｈｓｏ．ｓｒｃ　：ＰＣとＤＳＰの間で，ハンドシェークを行うプログラム

各プログラムをｒａ７７２３０にかける．

例：ｒａ７７２３０　a:プログラム名 [RET]

各プログラムは，次の様になる．

main1.src　→main1.prn,main1.rel

mains0.src →mains0.prn,mains0,prn

hso.src　→hs0.prn,hs0.rel

これらのプログラムをＬＫ７７２３０にかける．

例：ｌｋ７７２３０　a:main1 a:mains0 a:hs0 to a:main1 [RET]

すると　ａドライブにｍａｉｎ１．ｌｎｋ　とｍａｉｎ１．ｍａｐができる．

これをｏｃ７７２３０にかける．

例：ｏｃ７７２３０　a:main1 [RET]

これらの作業によって実行プログラム

ｍａｉｎ１．ｈｅｘ　ができる．

次に，ＰＣ－９８００側のプログラムを構造を示す．ＰＣ側のプログラムは４つあり，すべてＣ言語（コンパイラは，クイックＣを使用した．）で書かれている．

ｓｉｍｌｏａｄ．ｃ：シミュレーション用のデータをロードするプログラム

ｓｃｒｔｃｎｔ．ｃ：ロードされたデータの変更，確認するプログラム

ｄｓｐ＿ｈｓ．ｃ：ＰＣとＤＳＰの間で，ハンドシェークを行うプログラム

００１．ｃ：メインプログラム

各々を，コンパイルした後にリンクして，実行プログラム００１．ＥＸＥができる．

１）ＤＳＰにＤＳＰ用の実行プログラムをロードさせる．

例：ＭＤＲ　a:main1.hex[RET]

２）ＰＣ－９８００用の実行プログラムでコンピュータを作動させる．

ＤＳＰ用の実行プログラムを作成する際に，使用したプログラム（＊＊７７２３０）は，ＤＳＰボードにデッバッカ（TPREAD.C等）と共に付属されている．

4．2　実験結果

今回は，船舶の運動制御に適応的手法が有効であることを確認するための実船実験である．そのため比較的制御が簡単であると思われる一次遅れであるところの速度制御を試みた．速度制御の方向として各方向が考えられるが，現在の汐路丸の状態（例えば現在，ドップランナが不調である）から停船状態よりの旋回角速度を選んでみた．制御入力としては，バウ，スターン両スラスタが上げられるが，スターンスラスタは，ポンプ形式であるのでキャビテーションの心配がある．そこで，ＣＰＰ翼角に入力できる，バウスラスタからの制御入力を選んでみた．（ＣＰＰ翼角を制御するだけであるので，過電力を入力しないことだけ注意すれば良いが，過電力入力防止機構が汐路丸には，備わっているので実際には考慮する必要性は無く，スターンスラスタを使用するより安全である，と判断した）

又，本実験は船舶の運動制御に適応的手法が有効であることを確認することが実験目的であるので，今回の実験を基に様々なバージョンアップが考えられるので，目標値をリアルタイムで任意に変更できる等，設計にゆとりをもって設計してあり，実船実験においても目標値は任意に変化させてある．

Ｆｉｇ－４．１及びＦｉｇ－４．２は適応ゲイン（フ

ィードバック，フィードフォワード共に）を一定に固定して機器のテストを兼ねて実験してみたものである．これらの図例では，出力信号　入力信号が入出力の確認を始め，フィードバックがなされている事など確認できた．しかし，適応制御がなされていないので当然制御性は，悪くなっている．

Ｆｉｇ－４．３及びＦｉｇ－４．４は先の図例とは異なり，適応制御を施している．そのために，これらの図例には適応ゲインの変化が見られる．制御完了までに至ってないが，モデル出力に追従する方向にある．これらの図例では，適応スピードが小さいので適応スピードを上げることにより，制御性の向上が期待できる．
（目標値については，Ｆｉｇ－４．３では，１
Ｆｉｇ－４．４では，－１
の後任意変化させてある）

Ｆｉｇ－４．５及びＦｉｇ－４．６は，さらに適応スピードをあげてみた．制御完了と思われる時点まで到達してはいないが，おおむね制御する方向にあると思われる．ここまでの実験で適応スピードというパラメータの有効性が確認できる．（目標値は，任意変化させてある）

＊実験終了時点での適応ゲイン

	フィードバック	フィードフォワード
Fig-4.1	１．００(固定)	１．００(固定)
Fig-4.2	１．００(固定)	１．００(固定)
Fig-4.3	０．９１	０．５２
Fig-4.4	０．８３	０．５８
Fig-4.5	１．３７	１．１０
Fig-4.6	２．６３	０．２２

これらの図例では，時間，環境等（千葉県船形沖　台風避難の船舶が多数停泊していた）の良好な制御がなされるまでに至らなかったが，おおむね制御する方向にあることが，うかがえる．

実験結果の図例全ての実船旋回角速度出力にノイズが見られるが，これは実験当日（１９９０年１１月２９日）汐路丸の角速度出力端子が不調であったため，ジャイロからの現在方位の変化を微分して角速度としたためのノイズであると思われる．

また，旋回角速度の初期値が０から始められなかったのは，台風の影響で潮流や風等の外乱が思いの他大きく

船の初期設定を停船状態に保てなっかたためであり実験の主旨や，方法とは関係ない．

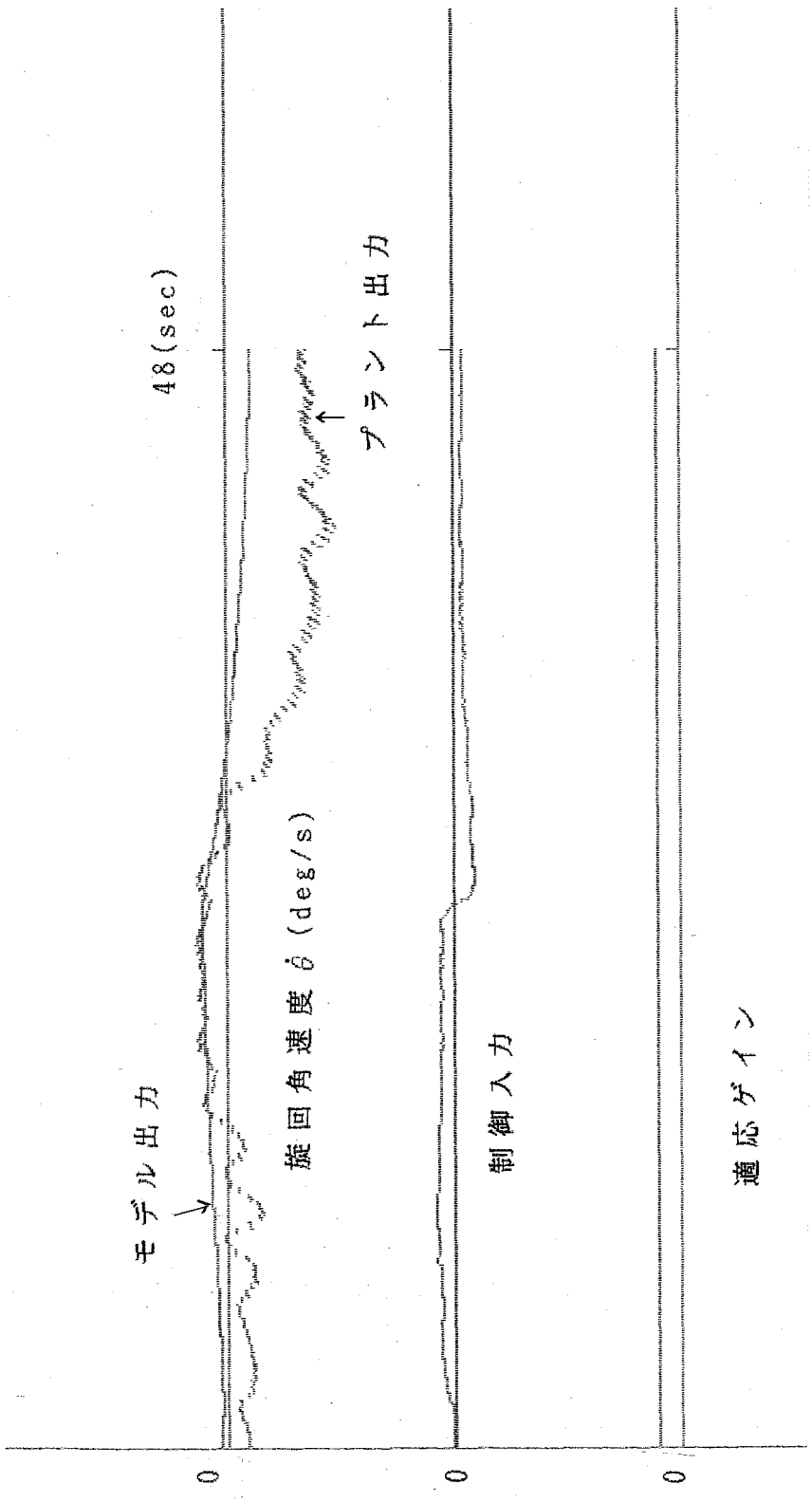


Fig-4.1

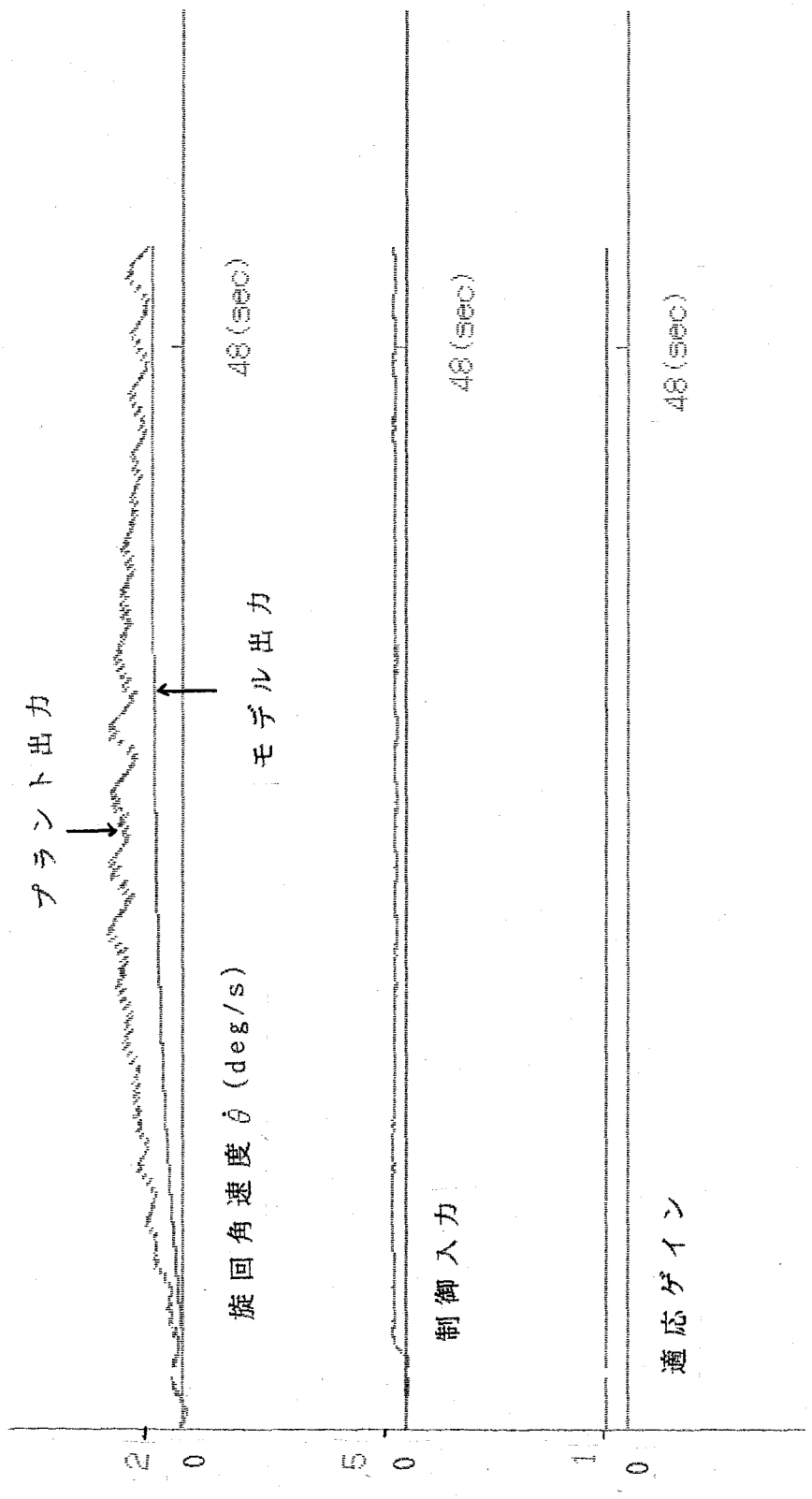
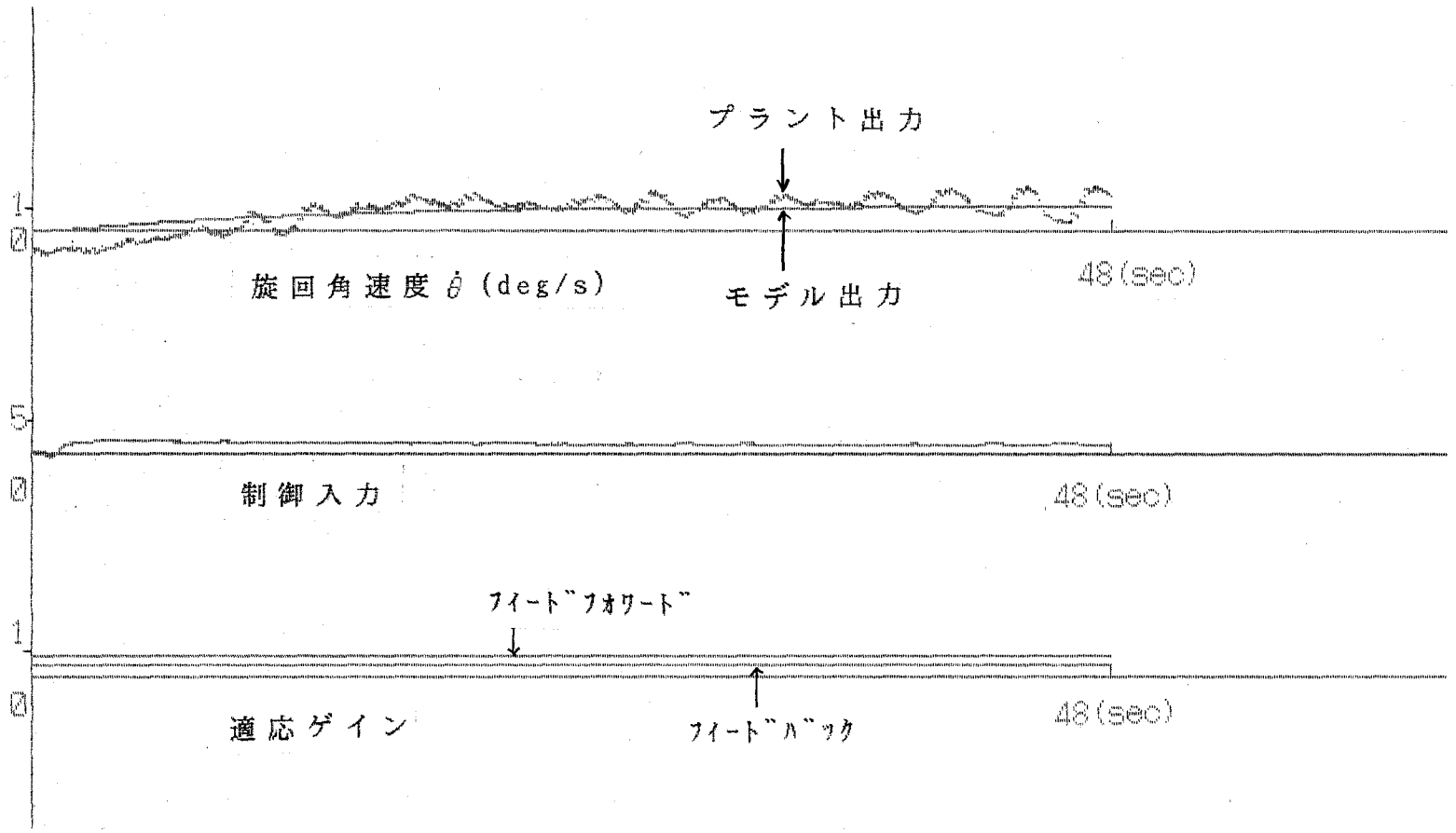

プラント出力
モデル出力
旋回角速度 $\dot{\theta}$ (deg/s)
制御入力
適応ゲイン
48(sec)
48(sec)
48(sec)
2
0
5
0
1
0

Fig-4.2



Fig－4．3　　適応スピード　0．05

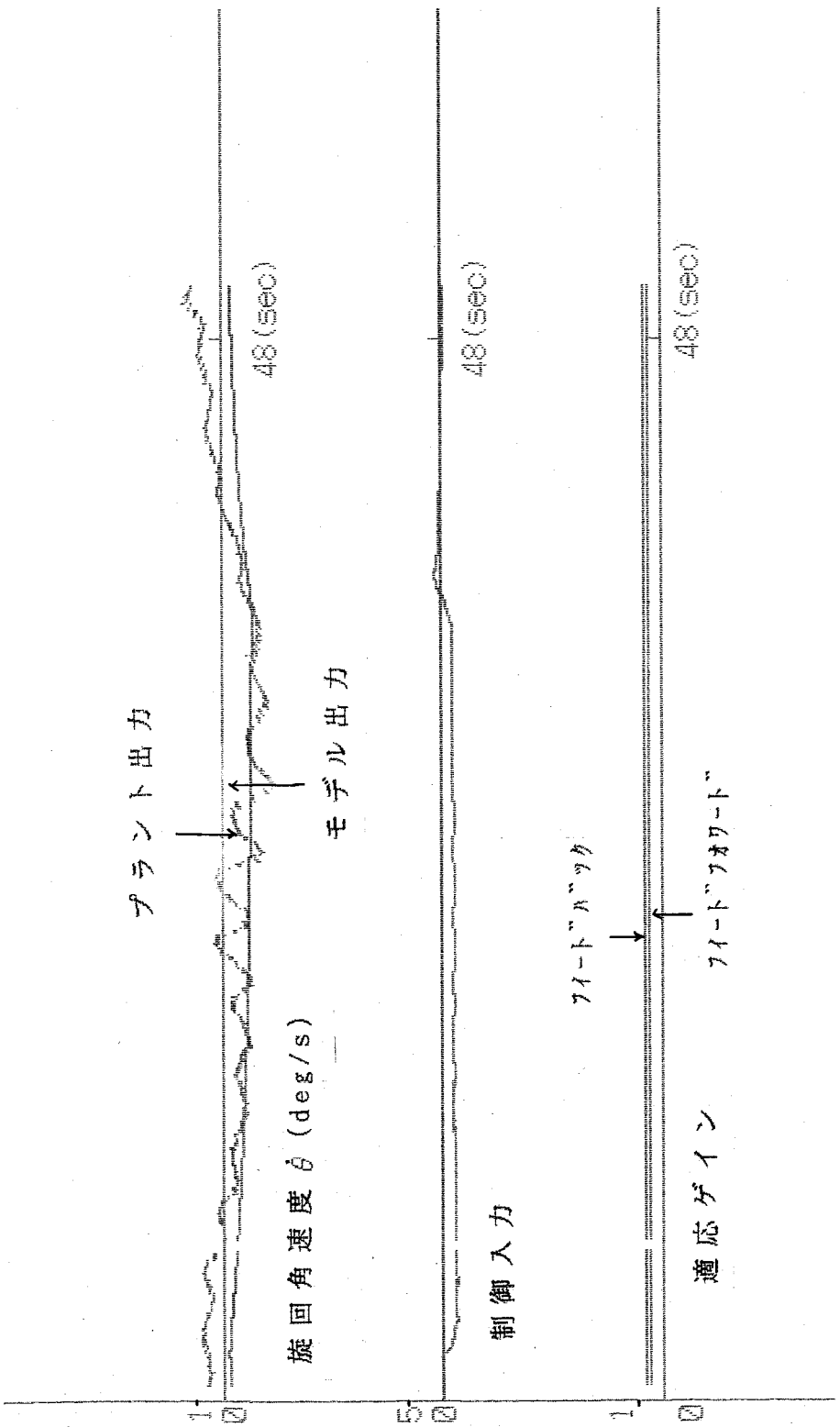


Fig-4.4 適応スピード 0.05

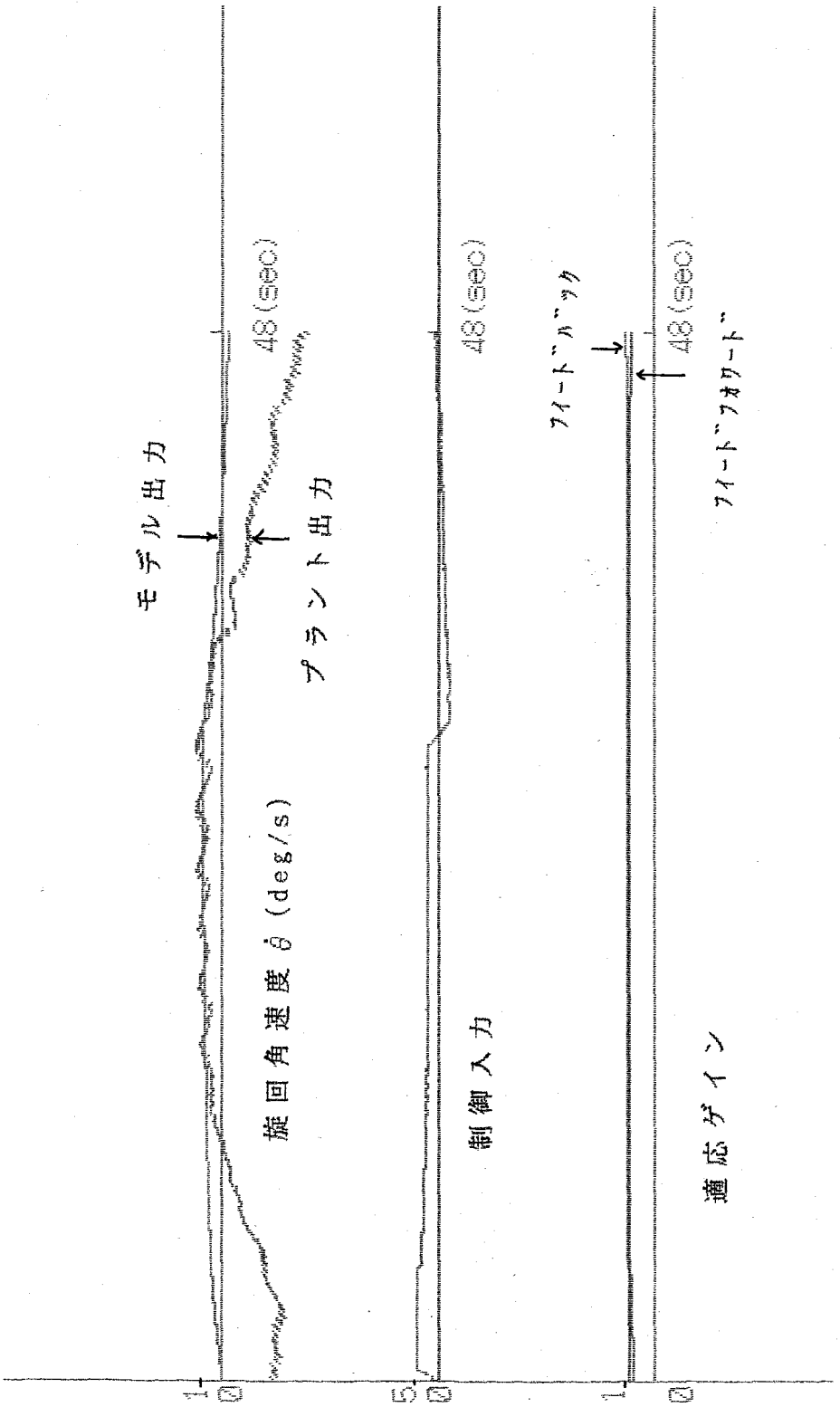


Fig-4.5 適応スピード 2.5

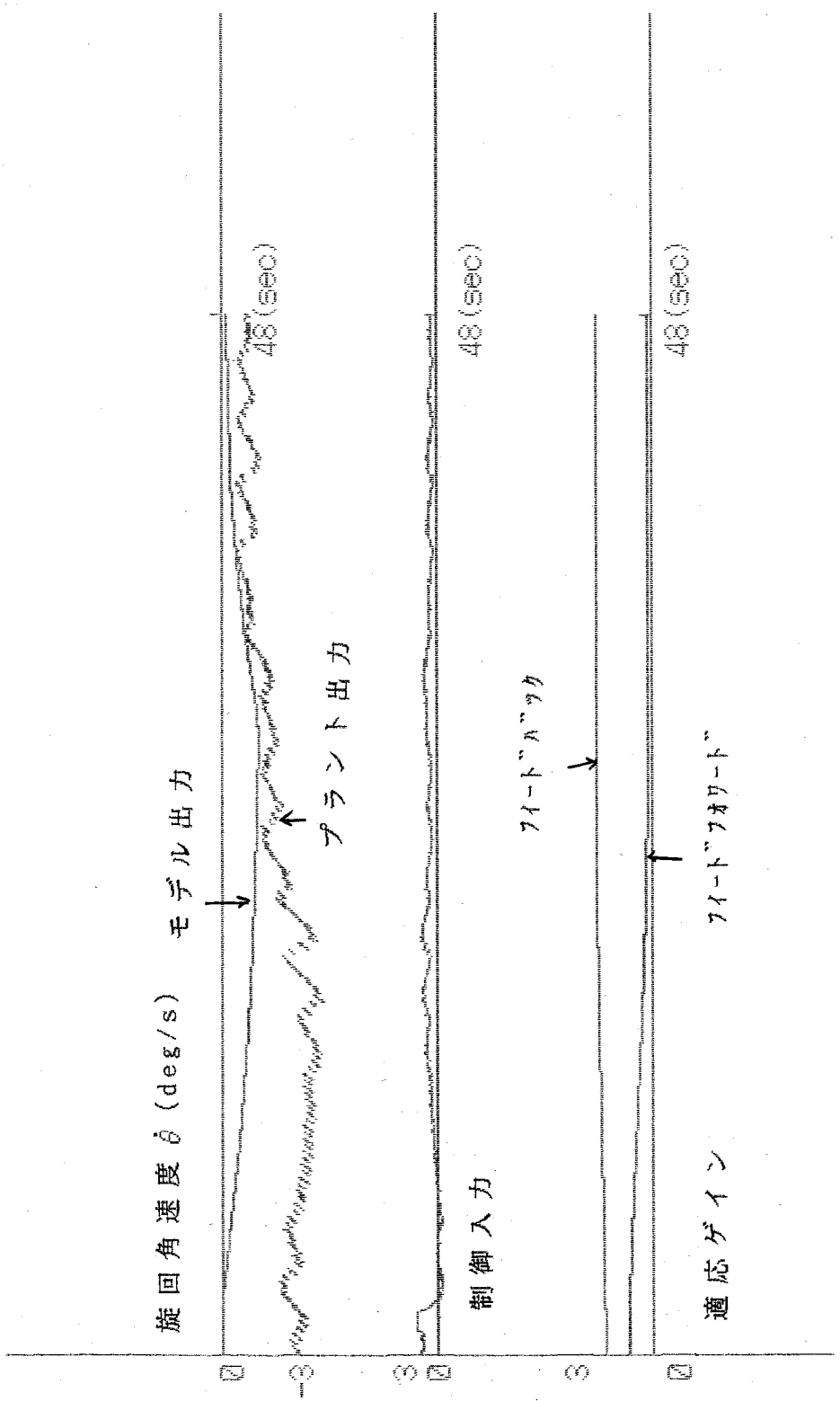


Fig-4.6 適応スピード 2.5

終わりに

今回の実船実験では，時間環境等の制約もあり良好な制御成績が得られるまでに至らなかった．しかし実験結果を見ても判る様におおむね制御する方向にあると思われる．

すなわち，適応的手法による船舶の運動制御が可能であると判断できるものと思われる．

さらに今回の実船実験において，以下の問題点が新に確認できた．

＊　スラスターの入力方向を表すフラッカーが，メインコンソール（ブリッジ，　機関制御室）にあるが，このフラッカーは入力方向に関係なく入力値が変化する際に，変化方向に点灯する．

例：入力値　0.1　->　0.2　プラス方向に点灯

入力値　-0.1　->　0.1　プラス方向に点灯

入力値　0.1　->　0.0　マイナス方向に点灯

入力値　0.1　->　-0.1　マイナス方向に点灯

頻繁な，フラッカー点灯はリレーの寿命そのものを短

くするだけではなく，船のシステム自体を痛める可能性も出てくる．

則ち，入力の際に数秒の時定数を持った一次遅れ関数などの，入力フィルター等を考える必要が出てくるが，より精密な，より即応性の高い制御を求める時に数秒の時定数を持った入力フィルターが，制御性に関わらないとは，言いがたいものがあると思われる．この問題は，制御性そのものに直接影響しており，見方によっては制御の本質的な問題でもあるので，慎重に取り扱う必要があるものと思われる．

また，今回の実船実験を基本実験として取扱，得られた適応ゲインを Tuned_gain （今回の実験では，Tuned_gain が得られるまでに至っていないと思われる）とみなして，規範モデルの再考慮を行うとこによって，さらなる制御性が期待できる．

今回の実船実験では，制御入力にバウスラスターからの入力のみで行ってみただけであり，船舶の統合的な制御を検証するまでには，まだまだ至っていない．また，実船実験を行うことで，陸上で考えているより，多くの問題点が露見されてくる．今後船舶の統合的な制御を考える際には，実船実験を多く伴い，問題点を発見し，着実に解決する事によって，適応的手法による統合的な船舶の運動制御の実現が期待できる．

謝辞

最後に実船実験では，汐路丸萩原船長，堀田機関長を始めとする乗り組みの方々に多くの御助言，御指導をいただき感謝いたします．

舶用制御工学科の森下教授，濱田助教授には，大学時代から長期にわたり，多くの御指導，御教授いただき感謝の意を表します．平素よりの，河野教授，平沼助教授，三嶋技官，大石技官の御助言に深く感謝します．

参考文献

（１）石川島播磨重工株式会社：東京商船大学練習船汐路丸操船ブックレット(1987)

（２）航海便覧編集委員会　：航海便覧 海文堂(1976)

（３）濱田　和恭　：適応制御系のロバスト性に関する研究 名古屋大学大学院工学部博士論文(1987)

（４）濱田，鈴木　：ロバストな適応制御則 24th Annual Allerton Conference on Commuication Control & Computing (1985)

（５）武藤　義美　：適応的手法による倒立振子姿勢制御 東京商船大学大学院修士論文(1989)

（６）安藤，高橋　：ＤＤＣによるディーゼル主機回転数制御についての研究

東京商船大学卒業論文
(1989)

(7) 武藤，濱田，森下 : 倒立振子のロバスト適応制御則
計測制御学会一論文集
(1990)

(8) 森下，藻垣 : パーソナルコンピュータを用いたモデル規範型適応制御のレベルサベージタンクへの応用
東京商船大学学術講演会論文集(1986)

＊ＤＳＰを使用したプログラム例

プログラム例としてＤＳＰを使ったシミュレーション用のメインプログラムを載せる．

アッセンブラで書かれた物は，ＤＳＰにロードするための，Ｃで書かれたものは，ＰＣ－９８用である．

```
        NAME    MAIN1.SRC       ;name of this program

;       ++++++++++++++++++++++++++++++++++++++

        RAMALCLR        EQU     0017H   ;
        SINWR0          EQU     0327H   ;
        SOUTWR0         EQU     0375H   ;
        FINWR0          EQU     0279H   ;
        FOUTWR0         EQU     02B0H   ;
        SINDR           EQU     032CH   ;
        SOUTDR          EQU     037BH   ;
        FOUTDR          EQU     02B4H   ;
        SINRAM0         EQU     0084H   ;
        SINRAM1         EQU     00B2H   ;
        SOUTRAM0        EQU     0399H   ;
        SOUTRAM1        EQU     03B9H   ;
        SQUAE           EQU     013BH   ;
        DIV             EQU     075CH   ;

;       ++++++++++++++++++++++++++++++++++++++

        PUBLIC  MAINI
        PUBLIC  MAIN

;       ++++++++++++++++++++++++++++++++++++++

        EXTRN   START_ID                ;
        EXTRN   MINTHI,MINTLOW          ;
        EXTRN   REQHLH                  ;
        EXTRN   BRRAM0,BRRAM1           ;
        EXTRN   BWRAM0,BWRAM1           ;
        EXTRN   DSPIEEE0,DSPIEEE1       ;
        EXTRN   IEEEDSP0,IEEEDSP1       ;
        EXTRN   FLTFIX0,FLTFIX1         ;
        EXTRN   FIXFLT0,FIXFLT1         ;
        EXTRN   RMOVS01,RMOVS10         ;
        EXTRN   RAMCLR0,RAMCLR1         ;

;       ++++++++++++++++++++++++++++++++++++++

        EXTRN   MULX_R0,MULX_R1         ;
        EXTRN   AD_ALGO                 ;
        EXTRN   AD_CONTL                ;
        EXTRN   E_FILTER                ;
        EXTRN   CK_LIMIT                ;
        EXTRN   LIMITER                 ;
        EXTRN   S_NORM1                 ;


;       ++++++++++++++++++++++++++++++++++++++
```

```
DINR0           EQU 0H          ;Ram0 addr. and val data
AM_IN           EQU 4           ;amount of input data
ADR_YAW_S       EQU DINR0+0H    ;I/O addr. of YAW_S(angle speed)
ADR_YAW_R       EQU DINR0+1H    ;I/O addr. of YAW_R
ADR_YOKO_S      EQU DINR0+2H    ;I/O addr. of YOKO_S
ADR_YOKO_R      EQU DINR0+3H    ;I/O addr. of YOKO_R

DOUTR0          EQU 10H         ;first addr. of out put
AM_OUT          EQU 14          ;amout of out put
ADR_UP          EQU DOUTR0+0H   ;I/O addr. of plant input(yaw)
ADR_UP_Y        EQU DOUTR0+1H   ;I/O addr. of plant input(yoko)
ADR_E           EQU DOUTR0+2H   ;I/O addr. of error(yaw)
ADR_E_Y         EQU DOUTR0+3H   ;I/O addr. of error(yoko)
ADR_YD          EQU DOUTR0+4H   ;I/O addr. of plant output(yaw)
ADR_YD_Y        EQU DOUTR0+5H   ;I/O addr. of plant output(yoko)
ADR_YR          EQU DOUTR0+6H   ;I/O addr. of model output(yaw)
ADR_YR_Y        EQU DOUTR0+7H   ;I/O addr. of model output(yoko)
ADR_ED          EQU DOUTR0+8H   ;I/O addr. of ED(yaw error after D(s)
ADR_ED_Y        EQU DOUTR0+9H   ;I/O adr. of EDY(yoko error after D(s
ADR_KY1         EQU DOUTR0+0AH  ;I/O addr. of KY1(yaw)
ADR_KR1         EQU DOUTR0+0BH  ;I/O addr. of KR1(yaw)
ADR_KY1_Y       EQU DOUTR0+0CH  ;I/O addr. of KY1Y(yoko)
ADR_KR1_Y       EQU DOUTR0+0DH  ;I/O addr. of KR1Y(yoko)

SADR_Z          EQU 30H         ;first addr. of regressor Z
AM_Z            EQU 2           ;amout of Z
SADR_ZY         EQU SADR_Z+0H   ;storing addr. of ZY
SADR_ZR         EQU SADR_Z+1H   ;storing addr. of ZR

SADR_Z_Y        EQU 35H         ;
AM_Z_Y          EQU 2           ;
SADR_ZY_Y       EQU SADR_Z_Y+0H ;
SADR_ZR_Y       EQU SADR_Z_Y+1H ;

MODELCOF        EQU 50H         ;first addr. of model coff
AM_MCOF         EQU 5           ;amount of model coff
ADR_M_COF1      EQU MODELCOF+0H ;

F_COF1          EQU 60H         ;first addr. of filter_cof1(ram0)
AM_COF1         EQU 10          ;amount of filter cof1
CCOMPCOF        EQU F_COF1+0H   ;first addr. of C(s) coff
DCOMPCOF        EQU F_COF1+5H   ;first addr. of D(s) coff

F_COF2          EQU DCOMPCOF+5  ;first addr. of filter_cof2
AM_COF2         EQU 10          ;amount of filter cof2
SF0_COF         EQU F_COF2+0H   ;first addr. of SF0 coff
SF1_COF         EQU SF0_COF+5H  ;first addr. of SF1 coff

                                ;Ram1 addr. and val data
CTRL            EQU 0H          ;first addr. of control data
AM_CTRL         EQU 5           ;amount of control and sts
CONTROL0        EQU CTRL+0H     ;addr. of control0
STS             EQU CTRL+AM_CTRL;firsr addr. of sts data
AM_STS          EQU 4           ;amount of sts data
STS0            EQU STS+0H      ;addr. of sts0
STS1            EQU STS+1H      ;addr. of sts1
STS2            EQU STS+2H      ;addr. of sts2
STS3            EQU STS+3H      ;addr. of sts3
```

```
UTLITY          EQU 20H          ;first addr. of utlity data
AM_UTIL         EQU 7            ;amount of utlity data
ADR_GDT         EQU UTLITY+0H    ;addr. of Gamma*dt
ADR_DT          EQU UTLITY+1H    ;addr. of dt
SCALE0          EQU UTLITY+2H    ;addr. of scaling factor 0 (0.1)
SCALE1          EQU UTLITY+3H    ;addr. of scaling factor 1 (0.01)
AMP_AD1         EQU UTLITY+4H    ;addr. of A/D converter amp G1
AMP_AD2         EQU UTLITY+5H    ;addr. of A/D converter amp G2
AMP_AD3         EQU UTLITY+6H    ;addr. of A/D converter amp G3

SADR_K          EQU 30H          ;first storing addr. of K
AM_K            EQU 2            ;amount of K
SADR_K_Y        EQU 35H          ;
AM_K_Y          EQU 2            ;

SADR_VAL        EQU 40H          ;storing addr. of val(Ram1)
SADR_E          EQU SADR_VAL+0H  ;storing addr. of E
SADR_E_Y        EQU SADR_VAL+1H  ;
SADR_EC         EQU SADR_VAL+2H  ;storing addr. of EC
SADR_EC_Y       EQU SADR_VAL+3H  ;
SADR_ED         EQU SADR_VAL+4H  ;sroring addr. of ED
SADR_ED_Y       EQU SADR_VAL+5H  ;
SADR_YD         EQU SADR_VAL+6H  ;storing addr. of YD
SADR_YD_Y       EQU SADR_VAL+7H  ;
SADR_YR         EQU SADR_VAL+8H  ;storing addr. of YR
SADR_YR_Y       EQU SADR_VAL+9H  ;
SADR_UP         EQU SADR_VAL+0AH;storing addr. of UP
SADR_UP_Y       EQU SADR_VAL+0BH;storing addr. of UPY(yoko)

MODELVAL        EQU 55H          ;first addr. of model val
SADR_M_VAL1     EQU MODELVAL+0H  ;storing addr. of actuater val
SADR_M_VAL2     EQU MODELVAL+5H  ;

F_VAL1          EQU 65H          ;first addr. of filter1 val
CCOMPVAL        EQU F_VAL1+0H    ;first addr. of C(s) val
DCOMPVAL1       EQU CCOMPVAL+5H  ;first addr. of D(s) val
DCOMPVAL2       EQU DCOMPVAL1+5H;
F_VAL2          EQU DCOMPVAL2+5H;first addr. of filter2 val
SF0_VAL         EQU F_VAL2+0H    ;first addr. of SF0 val
SF1_VAL         EQU SF0_VAL+5H   ;first addr. of SF1 val

                                 ;2-PORT ram address
TPR_DIN         EQU 1800H        ;first addr. of input data
TPR_DOUT        EQU 1810H        ;first addr. of output data
TPR_CTRL        EQU 1830H        ;first addr. of control
TPR_STS         EQU 1835H        ;first addr. of sts
TPR_UTIL        EQU 1850H        ;first addr. of utility data
TPR_MCOF        EQU 1860H        ;first addr. of model cof
TPR_INT_K       EQU 1870H        ;first addr. of int_k data
TPR_COF1        EQU 18C0H        ;first addr. of filter cof1 data
TPR_COF2        EQU 18D0H        ;first addr. of filter cof2 data
```

```
MIAN01          IMSEG   EXTERNAL        ;

;       +++++ MAINI ++++++++++++++++++++++++++

MAINI:                                  ;entry of this program
        MOV     TR,314A53H              ;TR <== 314A53H(ID data)
        CALL    START_ID                ;

        CALL    MINTLOW                 ;
        CALL    REQHLH                  ;

        CALL    RAMALCLR                ;

        CALL    MINTHI                  ;

        MOV     LC,AM_UTIL-1            ;load utility data
        MOV     AR,TPR_UTIL             ;from ex_addr. of TPR_UTIL
        MOV     IX1,UTLITY              ;to Ram1 addr.
        CALL    BRRAM1                  ;

        MOV     LC,AM_MCOF-1            ;load model_cof data
        MOV     AR,TPR_MCOF             ;from ex_addr. of TPR_MCOF
        MOV     IX0,MODELCOF            ;to Ram0 addr.
        CALL    BRRAM0                  ;

        MOV     LC,AM_K-1               ;load inital adaptive gain data
        MOV     AR,TPR_INT_K            ;from ex_addr. of TPR_INT_K
        MOV     IX1,SADR_K              ;to Ram1 addr.
        CALL    BRRAM1                  ;

        MOV     LC,AM_COF1-1            ;load C(s), D(s) coff data
        MOV     AR,TPR_COF1             ;from ex_addr. of TPR_COF1
        MOV     IX0,F_COF1              ;to Ram0 addr.
        CALL    BRRAM0                  ;

        MOV     LC,AM_COF2-1            ;load signal filter coff data
        MOV     AR,TPR_COF2             ;from ex_addr. of TPR_COF2
        MOV     IX0,SF0_COF             ;to ram0 addr.
        CALL    BRRAM0                  ;

        CALL    MINTLOW                 ;

        JMP     START_UP                ;

;       +++++ DATA I/O ++++++++++++++++++++++

MAINIO:                                 ;entry of this program
        CALL    REQHLH                  ;

        CALL    MINTHI                  ;

        MOV     LC,AM_IN-1              ;load input data
        MOV     AR,TPR_DIN              ;from ex_addr. of TPR_DIN
        MOV     IX0,DINR0               ;to Ram0 addr.
        CALL    BRRAM0                  ;

        MOV     LC,AM_CTRL-1            ;load control data
        MOV     AR,TPR_CTRL             ;from ex_addr. of TPR_CTRL
        MOV     IX1,CTRL                ;to Ram1 addr.
        CALL    BRRAM1                  ;
```

```
        MOV     LC,AM_OUT-1             ;write output data
        MOV     AR,TPR_DOUT             ;to ex_addr. of TPR_DOUT
        MOV     IX0,DOUTR0              ;from Ram0 addr.
        CALL    BWRAM0                  ;

        MOV     LC,AM_STS-1             ;write STS data
        MOV     AR,TPR_STS              ;to ex_addr. of TPR_STS
        MOV     IX1,STS                 ;from Ram1 addr.
        CALL    BWRAM1                  ;

        CALL    MINTLOW                 ;

        RET                             ;

;       +++++ START_UP ++++++++++++++++++++++

START_UP:                               ;entry of this program
        MOV     TR,315453H              ;TR <== 315453H (ID data)
        MOV     IX1,STS0                ;IX1 <== addr. of STS0
        MOV     [IX1],TR                ;STS0 <== TR

        CALL    MAINIO                  ;

                                        ;fixed data ==> floating data
        MOV     LC,AM_IN-1              ;all data exchange
        MOV     IX0,DINR0               ;IX0 <== first addr. of target
        CALL    FLTFIX0                 ;

                                        ;A/D converter amp.
        MOV     IX0,ADR_YAW_S           ;IX0 <== addr. of ADR_YAW_S
        MOV     IX1,AMP_AD1             ;IX1 <== addr. of AMP_AD1
        SPC     IX0                     ;
        MOV     LKR0,[IX1]              ;M <== yaw_s*G1
        MOV     WR2,M                   ;WR2 <== M
        NORM    WR2,TR                  ;normalizing by TR
        NORM    WR2,RD                  ;normalizing by RD
        MOV     IX0,ADR_YAW_S           ;IX0 <== addr. yaw_s
        MOV     [IX0],WR2               ;output for yaw_s

        MOV     IX0,ADR_YOKO_S          ;IX0 <== first addr. of yoko_s
        MOV     IX1,AMP_AD1             ;IX1 <== addr. of AMP_AD1
        SPC     IX0                     ;
        MOV     LKR0,[IX1]              ;M <== yoko_s*G1
        MOV     WR2,M                   ;WR2 <== M
        NORM    WR2,TR                  ;normalizing by TR
        NORM    WR2,RD                  ;normarizing by RD
        MOV     IX0,ADR_YOKO_S          ;IX0 <== addr. yoko_s
        MOV     [IX0],WR2               ;output for yoko_s

        MOV     IX0,ADR_YAW_R           ;IX0 <== addr. of ADR_YAW_R
        MOV     IX1,AMP_AD3             ;IX1 <== addr. of AMP_AD3
        SPC     IX0                     ;
        MOV     LKR0,[IX1]              ;M <== yaw_r*G3
        MOV     WR2,M                   ;WR2 <== M
        NORM    WR2,TR                  ;normalizing by TR
        NORM    WR2,RD                  ;normalizing by RD
        MOV     IX0,ADR_YAW_R           ;IX0 <== addr. yaw_r
        MOV     [IX0],WR2               ;output for yaw_r
```

```
        MOV     IX0,ADR_YOKO_R          ;IX0 <== addr. of yoko_s
        MOV     IX1,AMP_AD3             ;IX1 <== addr. of AMP_AD3
        SPC     IX0                     ;
        MOV     LKR0,[IX1]              ;M <== yoko_s*G3
        MOV     WR2,M                   ;WR2 <== M
        NORM    WR2,TR                  ;normarizing by TR
        NORM    WR2,RD                  ;normarizing by RD
        MOV     IX0,ADR_YOKO_R          ;IX0 <== addr. of yoko_s
        MOV     [IX0],WR2               ;output for yoko_r

ST_YD_YAW:                              ;filtering YAW_S
        MOV     IX0,ADR_YAW_S           ;IX0 <== addr. of YAW_S
        MOV     WR1,[IX0]               ;WR1 <== YAW_S
        MOV     IX0,ADR_YD              ;IX0 <== I/O addr. of YD
        MOV     IX1,SADR_YD             ;IX1 <== storing addr. of YD
        MOV     [IX0],WR1               ;Ram0 <== YD sending
        MOV     [IX1],WR1               ;Ram1 <== YD storing

ST_YD_YOKO:                             ;filtering YOKO_S
        MOV     IX0,ADR_YOKO_S          ;IX0 <== addr. of yoko_s
        MOV     WR1,[IX0]               ;WR1 <== yoko_s
        MOV     IX0,ADR_YD_Y            ;IX0 <== I/O addr. of YD_Y
        MOV     IX1,SADR_YD_Y           ;IX1 <== storing addr. of YD_Y
        MOV     [IX0],WR1               ;Ram0 <== YD_Y sending
        MOV     [IX1],WR1               ;Ram1 <== YD_Y storing

K_OUT_YAW:                              ;Ki val sending to DATAOUT(yaw)
        MOV     LC,AM_K-1               ;amount of Ki(yaw)
        MOV     IX0,ADR_KY1             ;IX0 <== first I/O addr. of Ki(yaw)
        MOV     IX1,SADR_K              ;IX1 <== first addr. of source(yaw)
        CALL    RMOVS01                 ;

K_OUT_YOKO:                             ;Ki val sending to dataout(yoko)
        MOV     LC,AM_K_Y-1             ;amount of Ki(yoko)
        MOV     IX0,ADR_KY1_Y           ;IX0 <== first addr. of Ki(yoko)
        MOV     IX1,SADR_K_Y            ;IX1 <== first addr. of source(yoko)
        CALL    RMOVS01                 ;

        MOV     LC,AM_K-1               ;amount of sending data(yaw)
        MOV     IX0,ADR_KY1             ;IX0 <== first addr. of Ki(yaw)
        MOV     BP1,SCALE0              ;BP1 <== addr. of scale0 for yaw
        CALL    MULX_R0                 ;

        MOV     LC,AM_K-1               ;amount of sending data(yoko)
        MOV     IX0,ADR_KY1_Y           ;IX0 <== first addr. of Ki(yoko)
        MOV     BP1,SCALE0              ;BP1 <== addr. of scale0 for yoko
        CALL    MULX_R0                 ;

                                        ;fixed point <== floating point
        MOV     LC,AM_OUT-1             ;amount of all data
        MOV     IX0,DOUTR0              ;IX0 <== first addr. of target
        CALL    FIXFLT0                 ;

        MOV     IX1,CONTROL0            ;IX1 <== control0
        MOV     WR1,000001H             ;chack data for control
        AND     WR1,[IX1]               ;control0 = 0 or not
        JNZ0    START_UP                ;if control0 =1 then go to start
                                        ;else go dawn
```

```
;       +++++ MAIN CONTROL ++++++++++++++++++
MAIN:                                   ;Main Adaptive Control entry
        MOV     TR,315443H              ;TR <== status data
        MOV     IX1,STS0                ;IX1 <== addr. of sts0
        MOV     [IX1],TR                ;sts0 <== TR

        CALL    MAINIO                  ;

                                        ;floating point <== fixed point
        MOV     LC,AM_IN-1              ;amount of all data
        MOV     IX0,DINR0               ;IX0 <== first addr. of target
        CALL    FLTFIX0                 ;

YAW_AMP:                                ;A/D comverter amp. for yaw
        MOV     IX0,ADR_YAW_S           ;IX0 <== addr. of yaw_s
        MOV     IX1,AMP_AD1             ;IX1 <== addr. of amp_ad1
        SPC     IX0                     ;
        MOV     LKR0,[IX1]              ;M <== yaw_s*G1
        MOV     WR2,M                   ;WR2 <== M
        NORM    WR2,TR                  ;normalizing by TR
        NORM    WR2,RD                  ;normalizing by RD
        MOV     IX0,ADR_YAW_S           ;IX0 <== yaw_s
        MOV     [IX0],WR2               ;output for yaw_s

YOKO_AMP:                               ;A/D comverter amp. for yoko
        MOV     IX0,ADR_YOKO_S          ;IX0 <== addr. of yoko_s
        MOV     IX1,AMP_AD1             ;IX1 <== addr. of amp_ad1
        SPC     IX0                     ;
        MOV     LKR0,[IX1]              ;M <== yoko_s*G1
        MOV     WR2,M                   ;WR2 <== M
        NORM    WR2,TR                  ;normarizing by TR
        NORM    WR2,RD                  ;normarizing by RD
        MOV     IX0,ADR_YOKO_S          ;IX0 <== yoko_s
        MOV     [IX0],WR2               ;output for yoko_s

YR_YAW:                                 ;model output YR_YAW
        MOV     IX0,ADR_YAW_R           ;IX0 <== addr. of yaw_r
        MOV     WR1,[IX0]               ;WR1 <== input
        MOV     IX0,ADR_M_COF1          ;IX0 <== addr. of model coff
        MOV     IX1,SADR_M_VAL1         ;IX1 <== addr. of model val
        MOV     BP1,ADR_DT              ;BP1 <== addr. of dt
        CALL    E_FILTER                ;
        MOV     IX0,ADR_YR              ;IX0 <== I/O addr. of YR
        MOV     [IX0],WR2               ;output for YR
        MOV     IX1,SADR_YR             ;IX1 <== storing addr. of YR
        MOV     [IX1],WR2               ;storing YR

YR_YOKO:                                ;model output for YR_YOKO
        MOV     IX0,ADR_YOKO_R          ;IX0 <== addr. of yoko_s
        MOV     WR1,[IX0]               ;WR1 <== input
        MOV     IX0,ADR_M_COF1          ;IX0 <== addr. of model coff
        MOV     IX1,SADR_M_VAL2         ;IX1 <== addr. of model val
        MOV     BP1,ADR_DT              ;BP1 <== addr. of dt
        CALL    E_FILTER                ;
        MOV     IX0,ADR_YR_Y            ;IX0 <== I/O addr. of YR_Y
        MOV     [IX0],WR2               ;output for YR_Y
        MOV     IX1,SADR_YR_Y           ;IX1 <== storing addr. of YR_Y
        MOV     [IX1],WR2               ;storing YR_Y
```

```
YD_YAW:                                 ;filtering of plant output(yaw)
        MOV     IX0,ADR_YAW_S           ;IX0 <== ADR_YAW_S
        MOV     WR1,[IX0]               ;WR1 <== ADR_YAW_S
        MOV     IX0,ADR_YD              ;IX0 <== I/O addr. of YD
        MOV     IX1,SADR_YD             ;IX1 <== storing addr. of YD
        MOV     [IX0],WR1               ;ADR_YD <== WR1
        MOV     [IX1],WR1               ;SADR_YD <== WR1

YD_YOKO:                                ;filtering of plant output(yoko)
        MOV     IX0,ADR_YOKO_S          ;Ix0 <== addr. of yoko_s
        MOV     WR1,[IX0]               ;WR1 <== yoko_s
        MOV     IX0,ADR_YD_Y            ;IX0 <== I/O addr. of YD_Y
        MOV     IX1,SADR_YD_Y           ;IX1 <== storing addr. of YD_Y
        MOV     [IX0],WR1               ;ADR_YD_Y <== WR1
        MOV     [IX1],WR1               ;SADR_YD_Y <== WR1

ZY_YAW:                                 ;cal of regresser KY(yaw)
        MOV     IX1,SADR_YD             ;IX1 <== storing addr. of YD
        MOV     WR1,[IX1]               ;WR1 <== YD
        MOV     IX0,SADR_ZY             ;IX0 <== storing addr. of ZY1
        MOV     [IX0],WR1               ;Ram0 <== ZY1

ZY_YOKO:                                ;call of regreeser KY_Y(yoko)
        MOV     IX1,SADR_YD_Y           ;IX1 <== storing addr. of YD_Y
        MOV     WR1,[IX1]               ;WR1 <== YD_Y
        MOV     IX0,SADR_ZY_Y           ;IX0 <== storing addr. of ZY_Y
        MOV     [IX0],WR1               ;Ram0 <== ZY1_Y

ZR_YAW:                                 ;cal of regresser KR(yaw)
        MOV     IX0,ADR_YAW_R           ;IX0 <== addr. of yaw_r
        MOV     WR1,[IX0]               ;WR1 <== yaw_r
        NEG     WR1                     ;ZR <== -yaw_r
        MOV     IX0,SADR_ZR             ;IX0 <== storing addr. of ZR1
        MOV     [IX0],WR1               ;Ram0 <== ZR1

ZR_YOKO:                                ;cal of regreesor KR_Y(yoko)
        MOV     IX0,ADR_YOKO_R          ;IX0 <== addr. of yoko_s
        MOV     WR1,[IX0]               ;WR1 <== yoko_r
        NEG     WR1                     ;ZR <== yoko_r
        MOV     IX0,SADR_ZR_Y           ;IX0 <== storing addr. of ZR_Y
        MOV     [IX0],WR1               ;Ram0 <== ZR_Y

E_YAW:                                  ;cal of error E
        MOV     IX1,SADR_YD             ;IX1 <== storing addr. of YD
        MOV     WR1,[IX1]               ;WR1 <== YD
        MOV     IX1,SADR_YR             ;IX1 <== storing addr. of YR
        MOV     WR2,[IX1]               ;WR2 <== YR
        SUBF    WR1,WR2                 ;WR1 <== WR1(YD)-WR2(YR)
        MOV     IX0,ADR_E               ;IX0 <== I/O addr. of E
        MOV     IX1,SADR_E              ;IX1 <== storing addr. of E
        MOV     [IX0],WR1               ;ADR_E <== E
        MOV     [IX1],WR1               ;Ram1 <== E
```

```
E_YOKO:                                 ;cal of error E_Y
        MOV     IX1,SADR_YD_Y           ;IX1 <== storing addr. of YD_Y
        MOV     WR1,[IX1]               ;WR1 <== YD_Y
        MOV     IX1,SADR_YR_Y           ;IX1 <== storing addr. of YR_Y
        MOV     WR2,[IX1]               ;WR2 <== YR_Y
        SUBF    WR1,WR2                 ;WR1 <== WR1(YD_Y) - WR2(TR_Y)
        MOV     IX0,ADR_E_Y             ;IX0 <== I/O addr. of E_Y
        MOV     IX1,SADR_E_Y            ;IX1 <== storing addr. of E_Y
        MOV     [IX0],WR1               ;ADR_E_Y <== E_Y
        MOV     [IX1],WR1               ;Ram1 <== E_Y

ED_YAW:                                 ;cal of ED
        MOV     IX1,SADR_E              ;IX1 <== storing addr. of E
        MOV     WR7,[IX1]               ;push to WR7
        MOV     IX1,SADR_UP             ;IX1 <== storing addr. of UP
        MOV     WR1,[IX1]               ;WR1 <== SADR_UP
        MOV     IX0,DCOMPCOF            ;IX0 <== first addr. of D(s) coff
        MOV     IX1,DCOMPVAL1           ;IX1 <== first addr. of D1(s) val
        MOV     BP1,ADR_DT              ;
        CALL    E_FILTER                ;
        MOV     WR1,WR7                 ;WR1 <== E
        ADDF    WR1,WR2                 ;WR1 <== WR1(E)+WR2(D(s)*UP)
        MOV     IX0,ADR_ED              ;IX0 <== I/O addr. of ED
        MOV     IX1,SADR_ED             ;IX1 <== storing addr. of ED
        MOV     [IX0],WR1               ;ADR_ED <== ED
        MOV     [IX1],WR1               ;SADR_ED <== ED

ED_YOKO:                                ;cal of E_Y
        MOV     IX1,SADR_E_Y            ;IX1 <== storing addr. of E_Y
        MOV     WR7,[IX1]               ;push to WR7
        MOV     IX1,SADR_UP_Y           ;IX1 <== storing addr. of UP_Y
        MOV     WR1,[IX1]               ;WR1 <== UP_Y
        MOV     IX0,DCOMPCOF            ;IX0 <== first addr. of D(s) coff
        MOV     IX1,DCOMPVAL2           ;IX1 <== first addr. of D2(s) val
        MOV     BP1,ADR_DT              ;
        CALL    E_FILTER                ;
        MOV     WR1,WR7                 ;WR1 <== WR7
        ADDF    WR1,WR2                 ;WR1 <== WR1(E_Y)+WR2(D(s)*UP)
        MOV     IX0,ADR_ED_Y            ;IX0 <== I/O addr. of ED_Y
        MOV     IX1,SADR_ED_Y           ;IX1 <== storing addr. of ED_Y
        MOV     [IX0],WR1               ;ADR_ED_Y <== ED_Y
        MOV     [IX1],WR1               ;SADR_ED_Y <== ED_Y

ADAPTIVE_YAW:                           ;entry of AD_ALGO
        MOV     IX0,ADR_ED              ;IX0 <== first addr. of ED
        MOV     WR1,[IX0]               ;WR1 <== ED
        MOV     LC,AM_K-1               ;amount of Ki,Zi
        MOV     IX0,SADR_Z              ;IX0 <== first addr. of regresser Zi
        MOV     IX1,SADR_K              ;IX1 <== first addr. of gain Ki
        MOV     BP1,ADR_GDT             ;

        CALL    AD_ALGO                 ;

        MOV     LC,AM_K-1               ;amount of Ki
        MOV     IX0,ADR_KY1             ;IX0 <== first addr. of Ki
        MOV     IX1,SADR_K              ;IX1 <== first addr. of Ki(souce)
        CALL    RMOVS01                 ;
```

```
                                        ;entry of AD_CONTL
        MOV     LC,AM_K-1               ;amount of Ki,Zi
        MOV     IX0,SADR_Z              ;IX0 <== SADR_Z
        MOV     IX1,SADR_K              ;IX1 <== SADR_K

        CALL    AD_CONTL                ;

        NEG     WR1                     ;WR1 <== -WR1

        MOV     IX0,ADR_UP              ;IX0 <== destination addr.
        MOV     [IX0],WR1               ;IX0 <== WR1(Zi*Ki)
        MOV     IX1,SADR_UP             ;IX1 <== storing addr. of UP
        MOV     [IX1],WR1               ;IX1 <== UP

ADAPTIVE_YOKO:                          ;entry of AD_ALGO
        MOV     IX0,ADR_ED_Y            ;ix0 <== addr. of ED_Y
        MOV     WR1,[IX0]               ;wr1 <== ED_Y
        MOV     LC,AM_K_Y-1             ;amount of K_Y
        MOV     IX0,SADR_Z_Y            ;IX0 <== storing addr. of Z_Y
        MOV     IX1,SADR_K_Y            ;IX1 <== storing addr. of Z_Y
        MOV     BP1,ADR_DT              ;

        CALL    AD_ALGO                 ;

        MOV     LC,AM_K_Y-1             ;amount of K_Y
        MOV     IX0,ADR_KY1_Y           ;IX0 <== addr. of KY1_Y
        MOV     IX1,SADR_K_Y            ;IX1 <== addr. of KY1_Y(source)
        CALL    RMOVS01                 ;

        MOV     LC,AM_K_Y-1             ;amount of K_Y
        MOV     IX0,SADR_Z_Y            ;IX0 <== addr. of Z_Y
        MOV     IX1,SADR_K_Y            ;IX1 <== addr. of K_Y

        CALL    AD_CONTL                ;

        NEG     WR1                     ;WR1 <== -WR1

        MOV     IX0,ADR_UP_Y            ;IX0 <== addr. of UP_Y
        MOV     [IX0],WR1               ;[IX0] <== UP_Y
        MOV     IX1,SADR_UP_Y           ;IX1 <== storing addr. of UP_Y
        MOV     [IX1],WR1               ;[IX1] <== UP_Y
```

```
CLS_STS1:                                   ;clear sts1
        CLR     WR2                         ;
        MOV     IX1,STS1                    ;IX1 <== sts1
        MOV     [IX1],WR2                   ;Ram1 <== 80_000000H(WR2)

K_YAW:                                      ;Ki val scaling for data out(yaw)
        MOV     LC,AM_K-1                   ;amount of Ki
        MOV     IX0,ADR_KY1                 ;IX0 <== first addr. of K
        MOV     BP1,SCALE0                  ;
        CALL    MULX_R0                     ;

K_YOKO:                                     ;Ki val scaling for data out(yoko)
        MOV     LC,AM_K_Y-1                 ;amount of Ki
        MOV     IX0,ADR_KY1_Y               ;IX0 <== first addr. of K
        MOV     BP1,SCALE0                  ;
        CALL    MULX_R0                     ;

        MOV     LC,AM_OUT-1                 ;amount of dataout
        MOV     IX0,DOUTR0                  ;IX0 <== dataout
        CALL    FIXFLT0                     ;

        JMP     MAIN                        ;

END
```

```
        /* --- Test program for simu --- */

#include <stdio.h>
#include <graph.h>
#include <math.h>
#include <dsp.h>
#include <m_start.h>
#include <beep.h>

        char    id_card[5];
        float   t=0.0;
        float   total_t=0.0;
        int     count=0;
        int     bcount, scount, limit;
        float   gscal_x;
        float   dt=0.02;
        float   yaw_r,yoko_r;
        float   cal_time;
        float   up, up_y, yd, yd_y, yr, yr_y, e, e_y;
        float   ky1, kr1, ky1_y, kr1_y;
        float   up_bs, up_ss;
        float   yaw_ar, yaw_sr, yawr;
        float   yoko_ar, yoko_sr, yokor;
        float   yaw_a, yaw_s, yaw;
        float   yoko_a, yoko_s, yoko;
        double  gyaw_s, gyaw, gyaw_r;
        double  gyoko_s, gyoko_r;

        unsigned short  status1, status2, status3;

        long    ad[8], da[16];
        long    control[5], status[4];

        extern  long    buffer_ctrl[1], buffer_sts[4];

main()
{
        char            hit;
        short           yon;
        short           error0, error1;

        void            b_line();
        void            display_graph();
        void            display_title();
        void            display_time();
        short           data_load();
        unsigned short  key_interruption(void);
        unsigned short  service(void);

/* --- Make data file  ---*/
```

```
        FILE    *fptri,*fptrd;
        char    *condition,*data;

        erase();
        printf(" データを保存するドライブとファイル名を入力して下さい ¥n");
        scanf("%s",data);

/* ------------------------*/

start:
        _setvideomode(_98RESS8COLOR);
        _settextrows(25);

        _clearscreen(_GCLEARSCREEN);
        _clearscreen(_GCLEARTEXT);
        _displaycursor(_GCURSORON);

        error0=dsp_start(id_card);              /* DSP start */

        data_input();                           /* Data Input for SImulation */

        mintlc();                               /* Initial data load */

                error1=data_load();

        reqhlh();

        control[0]=buffer_ctrl[0];              /* Inital DSP control data */

        erase();

/* --- Make condition file --- */

        printf("条件を保存するドライブとファイル名を入力して下さい ¥n");
        scanf("%s",condition);
        fptri = fopen(condition,"w");
        fprintf(fptri,"yaw_r    yoko_r   dt¥n");
        fprintf(fptri,"%f %f %f¥n",yaw_r, yoko_r,dt);
        fclose(fptri);

/* ---------------------------- */

ench:
        erase();                                /* clr for display */
        b_line();                               /* making b_line   */

        fptrd = fopen(data,"w");                /* data file open */

        limit = (10/dt);                        /* seculity time */

        do{
                count+=1;
                        if(23000 < count)       /* for mess of disk change */
                        {
                                warning();
                        }
                t+=dt;
                fprintf(fptrd,"%d %f %f %f %f %f¥n",count,yd,yd_y,ky1,kr1,ky1_y,kr1_y);
                swit_b(up_y);                   /* for b_security */
                swit_s(up);                     /* for s_security */
                real_plant(up_bs, up_ss);       /* for real plant */

                display_time(t, total_t);
                display_graph();
```

```
                at(2,20) ;printf("%f",yr);      /* modeloutput of yaw */
                at(3,20) ;printf("%f",yd);      /* realoutput of yaw */
                at(5,20) ;printf("%f",yr_y);    /* modeloutput of yoko */
                at(6,20) ;printf("%f",yd_y);    /* realoutput of yoko */
                at(10,30);printf("%f",up_bs);   /* output of B_T input */
                at(11,30);printf("%f",up_ss);   /* output of S_T input */
                at(3,50) ;printf("%f",yawr);    /* output of total_Yaw */

                        edsp_adc(0x7, ad);
                        edsp_dac(0x3, da);

                        control[0]=0;

                mintlc();
                        dsp_write(&ad[0], 0x1800, 4);
                        dsp_read(0x1810, &da[0] ,14);
                        dsp_write(control, 0x1830, 1);
                        dsp_read(0x1835, status, 4);
                reqhlh();

                status1=status[1];

                if (kbhit()){
                fclose(fptrd);
                        switch(service()){
                          case 1:    goto start;
                          case 2:    break;
                          case 3:    goto end;
                        }
                   }

if(t> cal_time-dt)
        enchou();
        b_line();

        }while(t < cal_time);

serv:
        switch(service()){
           case 1:      goto start;
           case 2:      goto ench;
        }
end:
        resetl();
        _clearscreen(_GCLEARSCREEN);
        _displaycursor(_GCURSORON);
        _setvideomode(_DEFAULTMODE);
        _exit();
}

/* --- security of b_thruster ------- */
swit_b(up_b)

        float   up_b;

{
        if( (0.0005 >  up_bs) && (-0.0005 < up_bs) )
        {
                bcount +=1;
                if( bcount == limit)
                {
                        bcount =0;
                        up_bs = up_b;
                }
                else
```

```
                        up_bs = 0;
                }
        }
        else
        {
                up_bs = up_b;
        }
}

/* --- security of s_thruster ------- */

swit_s(up_s)

        float   up_s;

{
        if( (0.000005 > up_ss) && (-0.000005 < up_ss))
        {
                scount +=1;
                if (scount == limit)
                {
                        scount =0;
                        up_ss = up_s;
                }
                else
                {
                        up_ss = 0;
                }
        }
        else
        {
                up_ss = up_s;
        }
}

/* --- For simu. plant --- */

real_plant(fb, fs)

float   fb, fs;
{
        if(yaw_sr>=0)
                yaw_ar=(fs-2190.57*yaw_sr*yaw_sr)/3165.65;
        else
                yaw_ar=(fs+2190.57*yaw_sr*yaw_sr)/3165.65;

        yaw_sr+=yaw_ar*dt;
        yawr+=yaw_sr*dt;

        yaw_s = yaw_sr*100; yaw = yawr;

}

/* --- Input for DSP --- */

edsp_adc(channel, f_pointer)

        short   channel;
        long    f_pointer;

{
        unsigned long   fix_float();

        ad[0]=fix_float(yaw_s);
        ad[1]=fix_float(yaw_r);
        ad[2]=fix_float(yoko_s);
```

```
        ad[3]=fix_float(yoko_y);

}

/* --- Out put form DSP --- */

edsp_dac(channel, f_pointer)

        short   channel;
        long    f_pointer;

{
        float   float_fix(unsigned long);

        up   =float_fix(da[0]);
        up_y =float_fix(da[1]);
        e    =float_fix(da[2]);
        e_y  =float_fix(da[3]);
        yd   =float_fix(da[4]);
        yd_y =float_fix(da[5]);
        yr   =float_fix(da[6]);
        yr_y =float_fix(da[7]);
        kyl  =float_fix(da[10]);
        krl  =float_fix(da[11]);
        kyl_y=float_fix(da[12]);
        krl_y=float_fix(da[13]);

}

unsigned long  fix_float(flvalue)

        float   flvalue;
{
        unsigned long   result, shiftdata;

        if (flvalue > 10.0)
                flvalue=10.0;
        if (flvalue < -10.0)
                flvalue=-10.0;


        shiftdata=flvalue*214748364;
        result=((shiftdata >> 8) & 0x00ffffff);

        return(result);
}

float   float_fix(fxvalue)

        unsigned long   fxvalue;
{
        float   result;
        long    shiftdata;

        shiftdata=((fxvalue & 0x00ffffff) << 8);
        result=shiftdata;
        result=result*4.65661289E-09;

        return(result);
}

/* --- For display --- */

void    display_time(time, total_time)

        float   time, total_time;
```

```
        at(24,65);printf("%#.3f ", time+total_time);
}

void    b_line()

{
        _displaycursor(_GCURSOROFF);
        at(0,0);printf(" Pause => ESC ");

        _moveto(0,100);_lineto(650,100);        _moveto(0,15);_lineto(0,450);
        _moveto(0,200);_lineto(650,200);
        _moveto(0,250);_lineto(650,250);
        _moveto(0,300);_lineto(650,300);

        at(2,10);printf("Yr_yaw  = ");
        at(3,10);printf("Y_yaw   = ");
        at(5,10);printf("Yr_yoko = ");
        at(6,10);printf("Y_yoko  = ");

        at(10,10);printf("バウスラスタ");
        at(11,10);printf("スターンスラスタ");
        at(3,30) ;printf("現在の回頭角は, ");
        at(24,73);printf("sec");

}

/* --- For dsplay graph --- */

void    display_graph()

{
        unsigned short time;
        unsigned short g_yd, g_yr,g_e,g_ydy, g_yry,g_ey;
        unsigned short g_kyl, g_krl, g_kyly, g_krly;

        time=t*gscal_x-15;

        g_yd =-yd*50+100;       /* yaw output of real */
        g_yr =-yr*50+100;       /* yaw output of model*/
        g_e  =-e*50+100;        /* yaw output of error*/
        pset(time, g_yd, 6); pset(time, g_yr, 2); pset(time, g_e, 3); /* yaw */

        g_ydy=-yd_y*50+200;     /* yoko output of real */
        g_yry=-yr_y*50+200;     /* yoko output of model*/
        g_ey =-e_y*50+200;      /* yoko output of error*/
        pset(time, g_ydy,6); pset(time, g_yry,2); pset(time,g_ey, 3); /* yoko*/

        g_kyl=-kyl*150+250;      /* gain output of yaw */
        g_krl=-krl*150+250;      /* gain output of yaw */
        pset(time, g_kyl, 5); pset(time, g_krl, 7);     /* yaw */

        g_kyly=-kyl_y*150+300;   /* gain output of yoko */
        g_krly=-krl_y*150+300;   /* gain output of yoko */
        pset(time, g_kyl, 5); pset(time, g_krly, 7);    /* yoko */
}

/* --- For interruption --- */

unsigned short service()

{
        unsigned short  yon;
        unsigned short  result;

what:
```

```
        _displaycursor(_GCURSORON);
        at(24, 0);printf(" Again(1), Cont(2), or END(3) ?  ");
        scanf("%d", &yon);

        switch(yon){
            case  1:    result=1;
                        break;
            case  2:    if (t >= cal_time)
                             enchou();
                        result=2;
                        break;
            case  3:    /*beep(0);*/
                         reset1(); _clearscreen(_GCLEARTEXT);
                        _clearscreen(_GCLEARSCREEN); _setvideomode(_DEFAULTMODE);
                        _exit();

            default:    goto what;
        }
out:
        _displaycursor(_GCURSOROFF);
         return(result);
}

enchou()
{
        total_t=total_t+t;
        t=0.0;
        _clearscreen(_GCLEARSCREEN);
}

warning()
{
        at(15,25);printf("ディスクの交換を考えて  ！！！");
        beep(1);
}

/* --- End of this program --- */
```